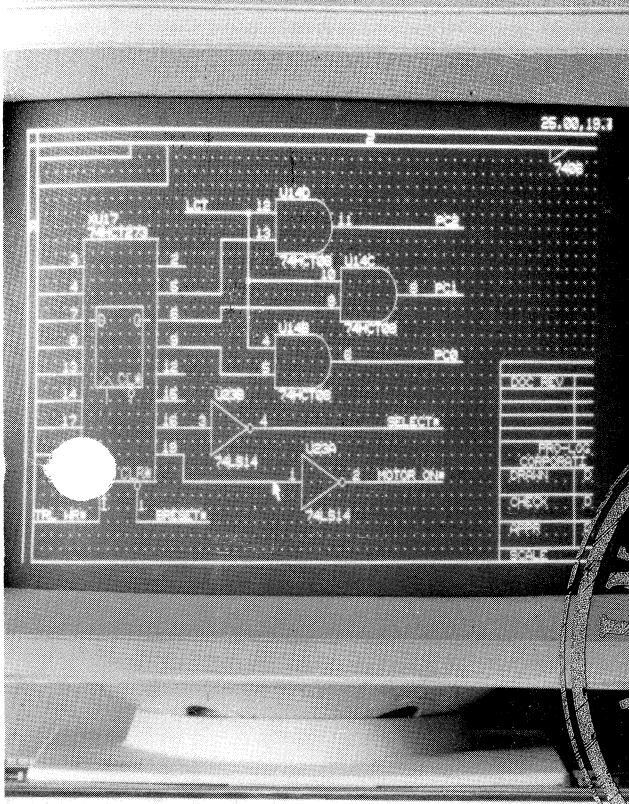
SYSTEM 2
USER'S GUIDE



INDUSTRIAL COMPUTERS
PRO-LOG
IN CONTROL
QUALITY

PRO-LOG® CORPORATION
113000-002 6/87

PRO-LOG CORPORATION
END-USER AGREEMENT

NOTICE: YOU SHOULD CAREFULLY READ THE FOLLOWING TERMS AND CONDITIONS REGARDING THE USE OF THE COMPUTER PROGRAMS INCLUDED WITH THIS PRODUCT. IF YOU DO NOT AGREE WITH THE FOLLOWING TERMS AND CONDITIONS, YOU SHOULD PROMPTLY RETURN THE PRODUCT INTACT AND YOUR MONEY WILL BE REFUNDED.

License

Pro-Log Corporation licenses the copyrighted computer software contained in this package, as follows:

1. You may use the program on a single machine.
2. You may copy the program for backup or modification purposes in connection with your use of the program on a single machine, so long as any copy contains Pro-Log's and Microsoft's copyright notices.
3. You may modify the program and/or merge it into another program for your use on the single machine, so long as any modified or merged program contains Pro-Log's and Microsoft's copyright notices.
4. You may transfer the program and license to another party if the other party agrees to accept the terms and conditions of this Agreement. If you transfer the program, you must at the same time either transfer all copies to the same party or destroy any copies not transferred, including all modifications and portions of the program contained or merged into other programs. Any such transfer automatically terminates your license to use the program.

Except as set forth above, you may not use, copy, modify or transfer the program, or any copy, modification or merged portion of it, in whole or in part.

From time to time, Pro-Log may advise you of corrections, enhancements, and updates to the computer program, which will be deemed to have been made upon the same terms and conditions as set forth in this Agreement.

Patents and Copyrights of Others

Pro-Log has no knowledge of any patent or copyright of others that is infringed by the computer program contained in this package. However, should there be made against you a legitimate claim by another of infringement of the other's United States patent rights or copyright by the computer program supplied under this order, Pro-Log will, at its own expense, defend any such claim and pay any award or damages assessed against you in any suit or proceeding based upon such a claim; provided, however, that Pro-Log is promptly notified by you of any such claim and is permitted to control any defense against the claim and its eventual resolution, and that you give Pro-Log all needed information, assistance and authority for Pro-Log to do so. In the event that your use of the program is enjoined by

a court as the result of such a claim, Pro-Log will, at its option and within a reasonable time, either secure for you the right to continue using that product by suspension of the injunction, by procuring for you a license or otherwise, or to replace the product with a non-infringing one, or to remove the enjoined Product and refund the sums paid by you for the program. These provisions, however, shall not apply to a claim based upon a modified or merged version of the program, in which case Pro-Log shall have no responsibility to you. The foregoing states Pro-Log's entire liability for infringement of copyrights or patents world-wide.

Limited Warranty

Pro-Log has made every reasonable effort to make sure that the program is accurate and free of errors but cannot guarantee this. However, Pro-Log does warrant for a period of ninety (90) days from the date of delivery to you, that the disk media containing the machine code or the computer program is electrically functional and readable. Pro-Log's liability shall be limited to replacing or repairing, at its option, any defective media which is returned F.O.B. Pro-Log's plant, but in no case is any such product to be returned without first obtaining the permission of Pro-Log. Products which have been subject to abuse, misuse, alteration, neglect, unauthorized repair or installation are not covered by warranty. Pro-Log shall make the final determination as to the existence and cause of any alleged defect.

THIS WARRANTY IS IN LIEU OF ANY OTHER WARRANTY EXPRESSED OR IMPLIED. PRO-LOG MAKES NO OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING ANY WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR OF MERCHANTABILITY. IN NO EVENT SHALL PRO-LOG BE LIABLE FOR INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES SUCH AS LOSS OF PROFITS OR BUSINESS OPPORTUNITIES, NOR FOR THE INABILITY OF THE CUSTOMER TO USE THE MACHINE READABLE CODE.

General Terms

You may not sublicense, assign or transfer the license or the program, except as expressly provided above.

This license may be terminated by you by destroying the program together with all copies, modifications and merged portions. It will also automatically terminate if you fail to comply with any term or condition of this Agreement.

This Agreement shall be governed by the laws of the state of California. You agree that these terms and conditions are the complete and exclusive statement of the agreement between Pro-Log and you with respect to the computer program contained in this package, and that they supersede any proposal or prior agreement, oral or written, that relate to such a program.

STD-DOS
SYSTEM 2
USER'S GUIDE

113000-002 250 6/87

FOREWORD

The information in this document is provided for reference only. Pro-Log does not assume any liability arising out of the application or use of the information or products described herein.

This document may contain or reference information and products protected by copyrights or patents and does not convey any license under the patent rights of Pro-Log, nor the rights of others.

Printed in U.S.A. Copyright 1987 by Pro-Log Corporation, Monterey, CA 93940. All rights reserved.

TRADEMARKS

BRIEF is a trademark of UnderWare Corporation. CROSSTALK is a registered trademark of Microstuf, Inc. DBASE is a trademark of Aston-Tate. Hercules Graphics is a trademark of Hercules Computer Technology Inc. IBM PC/XT/AT are registered trademarks of International Business Machines Corporation. Loopworks is a registered trademark of Equinox Data Corporation. Lotus is a registered trademark of Lotus Development Corporation. Microsoft Word, Microsoft Windows, and C Compiler are trademarks of Microsoft Corporation. Microsoft and MS-DOS 3.2 are registered trademarks of Microsoft Corporation. NEC V20 Microprocessor is a trademark of NEC Corporation. OrCAD is a trademark of OrCAD Systems Corporation. PAX is a registered trademark of Baker & Rabinowitz, Inc.

REFERENCES/BIBLIOGRAPHY

DOS Reference, Pro-Log document number 113091
DOS Technical Reference, Pro-Log document number 113092
Running MS-DOS, Van Wolverton, Microsoft Press
Advanced MS-DOS, Ray Duncan, Microsoft Press
Supercharging MS-DOS, Van Wolverton, Microsoft Press

CUSTOMER SUPPORT

For technical problems, please call Pro-Log Customer Support at (800) 538-9570 from 7:00 a.m. to 5:00 p.m. (Pacific Time).

CONTENTS

Page

SECTION 1. INSTALLATION AND OPERATION

Chapter 1. System 2 Overview 1-1

1.1 Introduction 1-1
1.2 More Information of the MS-DOS Operating System 1-2
1.3 About this User's Guide 1-2
1.3.1 Installation and Operation 1-2
1.3.2 Technical Reference 1-2
1.4 Additional User's Guides 1-3

Chapter 2. Inspecting Your System 2-1

2.1 Unpacking System 2 2-1
2.2 System Components and Factory-Installed Options 2-1
2.2.1 Model 10 2-1
2.2.2 Model 20 2-5

Chapter 3. Selecting an Operator Interface 3-1

3.1 Introduction 3-1
3.2 Interfacing Through a Monitor and Keyboard 3-5
3.2.1 Connecting a Monitor and Keyboard to System 2 3-6
3.3 Interfacing Through a Terminal or PC 3-6
3.4 Connecting a Terminal to System 2 3-7
3.5 Using a PC for System Control and Application Development 3-9
3.5.1 The PC as a Development System 3-9
3.5.2 PC Requirements 3-10
3.6 Connecting the PC to System 2 3-10
3.6.1 Using the 7888's RS-232-C Port 3-10
3.6.2 Using the RS-232-C Port of the 7312-1 or 7314A-1 3-11
3.7 The STD.EXE Terminal Emulation Utility 3-13
3.7.1 Using COM2 3-15
3.7.2 Printing the Screen 3-15

Chapter 4. System Operation 4-1

4.1 Booting System 2 from Floppy Disk 4-1
4.2 Booting System 2 from Hard Disk 4-2
4.3 Booting System 2 from ROM Disk 4-3
4.4 In Case of Difficulty 4-5
4.5 Preparing to Move System 2 4-5

Chapter 5. Installing Additional System Components 5-1

5.1 Introduction 5-1
5.2 System 2 Supports Up to 640K Bytes of System Memory 5-1
5.3 Customizing Your System I/O 5-3
5.3.1 I/O Options 5-3
5.3.2 I/O Mapping 5-4
5.3.3 DOS Devices 5-6
5.3.4 Additional I/O Facilities and Software Drivers 5-6
5.4 Interrupts 5-6

Chapter 6. DOS Devices 6-1

6.1 Character Devices 6-1
6.1.1 CON (Console) 6-1
6.1.2 COM1 6-3
6.1.3 SER1, SER2, SER3, and SER4 6-3
6.1.4 LPT1, LPT2, LPT3 6-5
6.1.5 @Clock 6-5
6.1.6 NUL 6-5
6.1.7 Modifying or Redirecting Character I/O 6-6
6.2 Block Devices 6-6
6.2.1 Virtual (Semiconductor) Disk 6-9
6.2.2 Mechanical (Floppy and Hard) Disk Drives 6-11
6.3 Installable Device Drivers 6-12

Chapter 7. Preparing Application Programs 7-1

7.1 Procedure 7-1

Chapter 8. Transferring Executable Files to a ROM Disk 8-1

8.1 The ROMDISK Utility Program 8-1
8.2 Downloading Your Binary or Hex Files to a PROM Programmer 8-6
8.2.1 The SENDFILE Utility Program 8-6

Chapter 9. Transferring Files Between System 2 and Other Systems 9-1

9.1 Additional System 2 Communications Utilities 9-1
9.2 Communications Using the System 2 Console Port 9-1
9.2.1 CROSSTALK Script and Command Files 9-1
9.2.2 System 2 Directly Connected to Another System 9-1
9.2.3 System 2 Remotely Connected to Another System Through a Modem 9-2
9.3 Communications Using the System 2 COM1 Port 9-4

SECTION 2. TECHNICAL REFERENCE

Page

Chapter 10. Hardware Reference 10-1

10.1 Introduction 10-1
10.2 IBM PC/XT Compatibility 10-1
10.3 Functional Description 10-1
10.3.1 System Specifications 10-1
10.3.2 System Components 10-2
10.4 Card Rack Dimensions and Power Connections 10-6
10.4.1 Card Rack Dimensions 10-7
10.4.2 Power Connections 10-12

Chapter 11. Software Reference 11-1

11.1 BIOS Functions 11-1
11.2 System Files 11-2
11.3 IO.SYS Initialization 11-4
11.3.1 Character Devices 11-4
11.3.2 Block Devices 11-5
11.4 Software Utilities 11-8
11.4.1 Microsoft Utilities 11-8
11.4.2 Pro-Log Hardware-Specific Utilities 11-8
11.4.3 Terminal Emulation Utility 11-9
11.4.4 Remote Operation of System 2 11-9
11.4.5 Utilities for Using COM1 to Transfer Files 11-10
11.4.6 ROM Disk Generation and Transfer Utility 11-10
11.5 Optional Software Packages 11-10
11.6 Documentation 11-11
11.6.1 Standard Documentation 11-11
11.6.2 Optional Documentation 11-11

Chapter 12. Technical Specifications 12-1

12.1 Specifications 12-1
12.1.1 General Specifications 12-1
12.1.2 Environmental Specifications 12-2
12.1.3 Mechanical Specifications 12-2
12.1.4 Electrical Specifications 12-3
12.1.5 System 2 Mapping 12-4

Chapter 13. Maintenance 13-1

13.1 Reference Drawings 13-1
13.2 Return for Repair/Warranty 13-1

Appendix A. BIOS Listing A-1

DOS | I/O SYS | I/O SYS | BIOS

ROM BOOT ROUTINE (BIOS)
DOS INITIALIZE
DOS FUNCTION (INTR 21H)
EXEC DEVICE DRIVER
MAKE REQ HEADER
DEVICE HEADER TABLE LINKED LIST
CON DEVICE HDR
AUX DEVICE HDR
PRN DEVICE HDR
CLOCK DEVICE HDR
COM1 DEVICE HDR
SER1 DEVICE HDR
SER2 DEVICE HDR
SER3 DEVICE HDR
SER4 DEVICE HDR
LPT1 DEVICE HDR
LPT2 DEVICE HDR
LPT3 DEVICE HDR
ROMDISK DEV HDR
MECH DISK HEADER
RAMDISK DEV HDR
STRATEGY ROUTINE
AUX/COM1 DRIVER
@@CON DRIVER
CON/SER1 DRIVER
SER2 DRIVER
SER3 DRIVER
SER4 DRIVER
EGA/KBD DRIVER
PRN/LPT1 DRIVER
LPT2 DRIVER
LPT3 DRIVER
CLOCK DRIVER
ROMDISK DRIVER
RAMDISK DRIVER
ERROR DRIVER
REQUEST HEADER
MECH DISK STRATEGY
ENTRY ROUTINE
COMMAND DISPATCH TABLE
COM1 CMD TABLE
@@CON CMD TABLE
SERX CMD TABLE
EGA/KBD CMD TABLE
LPTX CMD TABLE
CLOCK CMD TABLE
ROMDISK CMD TABLE
RAMDISK CMD TABLE
ERROR CMD TABLE
MECH DISK DRIVER
I/O INITIALIZE
CLOCK MANAGER
STD @@XFER@@ APPLICATION
@@CON MANAGER
COM MANAGER
SCC MANAGER
EGA/KBD MANAGER
PRINTER MANAGER
ROMDISK MANAGER
RAMDISK MANAGER
MECH DISK SERVICE (INTR 78)
HARDWARE INIT
CTC
TIME TICK (INTR 08H)
CLOCK SERVICE (INTR 1AH)
USER
TRANSFER SERVICE (INTR 72)
COM SERVICE (INTR 14H)
COM HDWR SERVICE (INTR 0CH)
SERIAL COM CTRL (INTR 71H)
SCC HDWR SERVICE (INTR 80-8FH)
USER
VIDEO SERVICE (INTR 10H)
TTY SERVICE (INTR 29H)
KEYBOARD SERVICE (INTR 16H)
KBD HDWR SERVICE (INTR 09H)
PRINTER SERVICE (INTR 17H)
MEMORY DISK (INTR 77H)
USER
DISK SERVICE (INTR 13H)
SCSI SERVICE (INTR 79)
SYSTEM CLOCK
7868 CPU CARD
TIME OF DAY SMARTWATCH
8254 CTC
8259 PIC
8250 UART
KEYBOARD BUFFER
7312/14 SCC CARD
SCC1 CTRL REG
SCC2 CTRL REG
SCC3 CTRL REG
SCC4 CTRL REG
SCC1 BUFFER
SCC2 BUFFER
SCC3 BUFFER
SCC4 BUFFER
7350 EGA/KBD CARD
DISPLAY MEMORY
DISP CTRL REGS
KEYBOARD CTRL
KEYBOARD BUFFER
7362 PRINTER CARD
PRINTER CTRL
7713/14/15 MEMDISK CARD
MEMORY DISK
7391 FLOPPY CARD
FLOPPY DISK CTRL
7390 SCSI CARD
SCSI CONTROLLER

MARCH 19, 1987

FILE: STDOS20.SCH

SIZE	DOC NO.	REV	SH OF
D			

106967A

Chapter 1

System 2 Overview

1.1 INTRODUCTION

The System 2 Industrial Computer is an STD BUS microcomputer that includes a fully licensed MS-DOS 3.2 operating system. Because the operating system of the IBM PC, PC/XT, and AT is based on Microsoft's MS-DOS operating system, System 2 provides a software environment compatible to the PC, yet in industrially robust hardware.

System 2 also provides many of the hardware features of the IBM PC/XT. It contains:

- A V20 CPU (80C88-compatible)
- 128K bytes of CMOS static RAM for system memory
- Battery-backed Time-of-Day clock
- PC-compatible RS-232-C serial port, interrupt controller and counter/timer

Depending upon the model and options you purchased, System 2 can also include the following hardware features:

- 3.5- or 5.25-inch floppy disk drives
- 3.5-inch hard disk drive
- Solid-state RAM and ROM disks
- EGA/CGA/Monochrome monitor and PC/XT keyboard interface
- 640K bytes of CMOS static system RAM
- 8087 math coprocessor
- Parallel printer interface
- Triple-voltage power supply

The combination of two powerful industry standards, the MS-DOS operating system and STD BUS hardware, is the basis of all Pro-Log STD-DOS systems. Pro-Log's System 2 uses the STD-DOS 2.X ROM BIOS to interface the operating system to the STD BUS cards contained in the system.

Where possible, CMOS devices were used to decrease the power consumption and extend the operating temperature range of the cards in the system. However, the STD BUS cards in the system present TTL levels to the bus and should not be mixed with CMOS STD BUS cards.

1.2 MORE INFORMATION ON MS-DOS

Because MS-DOS is an industry-standard operating system, there are hundreds of books available as learning resources. Our *DOS Reference* manual provides information on all the MS-DOS commands. If you are unfamiliar with the MS-DOS operating system, we found *Running MS-DOS* by Van Wolverton, published by Microsoft Press, to be helpful.

For a more thorough understanding of the MS-DOS operating system, our *DOS Technical Reference* (optional) provides more detailed information on the MS-DOS operating system, MS-DOS system calls, and device drivers. We also recommend *Advanced MS-DOS* by Ray Duncan and *Supercharging MS-DOS* by Van Wolverton, both published by Microsoft Press.

1.3 ABOUT THIS USER'S GUIDE

This user's guide is divided into two sections:

1.3.1 Installation and Operation

The first section, *Installation and Operation*, consists of Chapters 1 through 9. This section describes the basic features and operation of System 2. It provides information on how to connect the system to a PC or terminal for system control, how to power-on and boot the operating system, and how to transfer files between System 2 and PC (either directly or through a modem).

Chapters 5 and 6 provide information on how to expand the memory and I/O resources in your system. Chapters 7 and 8 discuss the procedure for developing an executable application program and transferring the program to a ROM disk drive (if your programs are to reside on a ROM disk drive). Chapter 9 provides information on how to transfer files between System 2 and other systems.

1.3.2 Technical Reference

The *Technical Reference* section consists of Chapters 10 through 13. Chapters 10 and 11 include a discussion of IBM PC/XT-compatibility and provide a description of system hardware and software components. The general, environmental, mechanical, and electrical specifications are provided in Chapter 12. Chapter 13 includes instructions for warranty repair services. The BIOS listing is provided in Appendix A.

1.4 ADDITIONAL USER'S GUIDES

Two (or more) additional user's guides have also been shipped with your system. They contain technical information about the STD BUS cards and accessories included with the system. The additional user's guides can be placed in the Hardware Reference binder provided.

The user's guides for the 7888 CPU Card and 7713/7714/7715 Solid-State Disk Cards are included if you ordered Model 10. If you ordered Model 20, the user's guides for the 7888 CPU Card and the 7391 Floppy Disk Controller Card are supplied.

Chapter 2

Inspecting Your System

2.1 UNPACKING SYSTEM 2

Although System 2 has been fully integrated and tested before shipment, you should take a few minutes to inspect the system before applying power. We recommend that, when possible, you begin with a check of the system as shipped before installing other optional components.

1. Inspect the system for damage that may have occurred during shipment.
2. Ensure that the cards are installed securely in the motherboard. The retaining bar should be locked in place by the two thumb screws on each side of the card rack.
3. Verify that all the system components and factory-installed options have been installed. System components vary according to the model and options you specified at the time of purchase. Model 10 and Model 20 system components and factory-installed options are described in the following pages.

All boards installed in the system can be identified by the model number stamped on the white ejector tab at the top of the card, just below the retaining bar. Revision letters and dash numbers on the ejector ensure that you can order spare and replacement parts that match your system configuration.

The system components and factory-installed options for Model 20 are listed on page 2-5. The system components and factory-installed options for Model 10 are listed below.

2.2 SYSTEM COMPONENTS AND FACTORY-INSTALLED OPTIONS

2.2.1 Model 10

All configurations of Model 10 include the following system components:

7888-01/03	Processor Card
7713-01/02/03/04	System Disk Card
7909A	Terminator Card
BX-Series	STD BUS card rack
RC713	Null Modem
RC716	RS-232-C Interconnect Cable
6001	STD-DOS System 2 User's Guide MS-DOS 3.2 Operating System Disk (5.25-inch) STD-DOS Utilities Disk (5.25-inch) Hardware Reference binder
6002	DOS Reference manual

CARD SLOTS AVAILABLE FOR ADDITIONAL STD BUS CARDS

Options 001 through 010:

Model 10, without options, provides three user card slots for installing additional STD BUS cards in the system. Options 001 through 008 increase the number of user slots (6 to 23) available in the system. If you ordered Option 008, your system can be mounted in a 19-inch RETMA rack. Option 010 configures the card rack for panel mounting so that System 2 can be easily secured to vertical surfaces.

Option 001:	6 user card slots
Option 002:	9 user card slots
Option 003:	12 user card slots
Option 004:	15 user card slots
Option 005:	18 user card slots
Option 006:	21 user card slots
Option 007:	23 user card slots
Option 008:	19-inch RETMA rack (provides 23 user card slots, 18 user slots with hard disk option)
Option 010:	Panel mount version of card rack

Some of the user card slots may be used by factory-installed options. Verify that the system you ordered has the correct number of user card slots and that the following components have been installed for any options you have purchased.

MODEL 10 DISK DRIVE OPTIONS

Option 020:

The capacity for a 256K byte user-supplied ROM Disk Drive B is replaced by a factory-installed 128K byte battery-backed static RAM Disk Drive B. The 7713-01 is replaced by the 7713-02.

Option 021:

A Winchester disk drive (20-megabyte capacity), 7390-02 SCSI Host Adapter Card, and power/interconnect cables are factory-installed in the card rack of the system. Occupies one user card slot.

Option 025:

The 7713-01, shipped with the standard configuration of Model 10, is replaced by the 7713-03. The 7713-03 provides the MS-DOS 3.2 operating system on ROM disk drive H and a 256K-byte user-supplied ROM disk drive I. This allows you to add external 3.5-inch or 5.25-inch floppy drives accessed as drives A and B. Because the 7713-01 is replaced by the 7713-03, no additional card slots are used when this option is ordered.

Option 026:

The 7713-01, shipped with the standard configuration of Model 10, is replaced by the 7713-04. The 7713-04 provides the MS-DOS 3.2 operating system on ROM disk drive H and a 128K-byte battery-backed RAM disk drive I. This allows you to add external 3.5-inch or 5.25-inch floppy drives accessed as drives A and B. Because the 7713-01 is replaced by the 7713-04, no additional card slots are used when this option is ordered.

POWER SUPPLY OPTIONS

Option 030:

The M281 Triple-voltage power supply is shipped with the system. This power supply can be user-configured for 115V or 230V operation.

Option 031:

The M283 Triple-voltage power supply is configured for 115V operation and shipped with the system.

Option 032:

The M283 Triple-voltage power supply is configured for 230V operation and shipped with the system.

Option 033:
The M284 Triple-voltage power supply is configured for 115V operation and shipped with the system.

Option 034:
The M284 Triple-voltage power supply is configured for 230V operation and shipped with the system.

SYSTEM MEMORY EXPANSION CARDS

Option 040:
A 7716-01 256K-Byte System RAM Card is installed to increase system memory to 384K bytes total. Occupies one user card slot.

Option 041:
The 7716-01 and 7716-02 256K-Byte System RAM Cards are installed to increase system memory to 640K bytes total. Occupies two user card slots.

OPERATOR INTERFACE CARDS

Option 050:
The 7350-01 EGA/Keyboard Card is installed in the system. Occupies one user card slot.

Option 051:
The 7314A-1 Quad RS-232-C Communications Card is installed in the system. Occupies one user card slot.

Option 052:
The 7312-1 Dual RS-232-C Communications Card is installed in the system. Occupies one user card slot.

COPROCESSOR CARD

Option 060:
The 7888-03 replaces the 7888-01. The 7888-03 includes the 8087 Coprocessor Card. The 8087 Coprocessor Card occupies one user card slot.

PARALLEL PRINTER INTERFACE

Option 061:
The 7392-01 Parallel Printer Interface Card is installed in the system. Occupies one user card slot.

Now that you have checked the system's factory installed options, Chapter 3 will show you how to connect the system to a monitor, terminal, or PC.

2.2.2 Model 20

All configurations of Model 20 include the following system components:

7888-01/03	Processor Card
7391-01	Floppy Disk Controller Card
903185	3.5-inch Floppy Disk Drive
113071	3.5-inch Floppy Disk Controller Cable
113068	3.5-inch Floppy Disk Power Connector
7909A	Terminator Card
BX-Series	STD BUS card rack
RC713	Null Modem
RC716	RS-232-C Interconnect Cable
6001	STD-DOS System 2 User's Guide
	MS-DOS 3.2 Operating System Disk (3.5-inch)
	STD-DOS Utilities Disk (5.25-inch)
	Hardware Reference binder
6002	DOS Reference manual

CARD SLOTS AVAILABLE FOR ADDITIONAL STD BUS CARDS

Model 20, without options, provides 6 expansion card slots for installing additional STD BUS cards in the system. Options 001 through 005 increase the number of expansion slots (9 to 18) available in the system. If you ordered Option 005, your system can be mounted in a 19-inch RETMA rack. Option 010 configures the card rack for panel mounting so that System 2 can be easily secured to vertical surfaces.

Options 001 through 010:

Option 001:	9 user card slots
Option 002:	12 user card slots
Option 003:	15 user card slots
Option 004:	18 user card slots
Option 005:	19-inch RETMA rack (provides 18 user card slots, 15 user slots with hard disk option)
Option 010:	Panel mount version of card rack

Some of the user card slots may be used by factory-installed options. Verify that the system you ordered has the correct number of user card slots and that the following components have been installed for any options you have purchased.

MODEL 20 DISK DRIVE OPTIONS

Option 024:
A Winchester disk drive (20-megabyte capacity), 7390-02 SCSI Host Adapter Card, and power/interconnect cables are factory-installed in the system's card rack. Occupies one user card slot.

Option 025:
The 7713-03 is added to Model 20. The 7713-03 provides the MS-DOS 3.2 operating system on ROM disk drive H and a 256K-byte user-supplied ROM disk drive I. Occupies one user card slot.

Option 026:
The 7713-04 is added to Model 20. The 7713-04 provides the MS-DOS 3.2 operating system on ROM disk drive H and a 128K-byte battery-backed RAM disk drive I. Occupies one user card slot.

POWER SUPPLY OPTIONS

Option 030:
The M281 Triple-voltage power supply is shipped with the system.

Option 031:
The M283 Triple-voltage power supply is configured for 115V operation and shipped with the system.

Option 032:
The M283 Triple-voltage power supply is configured for 230V operation and shipped with the system.

Option 033:
The M284 Triple-voltage power supply is configured for 115V operation and shipped with the system.

Option 034:
The M284 Triple-voltage power supply is configured for 230V operation and shipped with the system.

SYSTEM MEMORY EXPANSION CARDS

Option 040:
A 7716-01 256K-Byte System RAM Card is installed to increase your system memory to 384K bytes total. Occupies one user card slot.

Option 041:
The 7716-01 and 7716-02 256K-Byte System RAM Cards are installed to increase your system memory to 640K bytes total. Occupies two user card slots.

OPERATOR INTERFACE CARDS

Option 050:
The 7350-01 EGA/Keyboard Card is installed in the system. Occupies one user card slot.

Option 051:
The 7314A-1 Quad RS-232-C Communications Card is installed in the system. Occupies one user card slot.

Option 052:
The 7312-1 Dual RS-232-C Communications Card is installed in the system. Occupies one user card slot.

COPROCESSOR BOARD

Option 060:
The 7888-03 replaces the 7888-01. The 7888-03 includes the 8087 Coprocessor Board. The 8087 Coprocessor Card occupies one user card slot.

PARALLEL PRINTER INTERFACE

Option 061:
The 7392-01 Parallel Printer Interface Card is installed in the system. Occupies one user card slot.

Now that you have checked the system's factory installed options, Chapter 3 will show you how to connect the system to a monitor, terminal, or PC.

Chapter 3

Selecting an Operator Interface

3.1 INTRODUCTION

Although System 2 can execute programs from power-up, in most cases you will want to establish an operator interface initially.

The Console port (or Console device as it is termed by DOS) is the MS-DOS standard input and output device for operator interface. It is both an input (keyboard) and output (display) device. The Console is used to execute DOS commands, load and execute application programs, enter system parameters to the application software, or display system status.

The 7350-01 EGA/Keyboard Card and the 7312-1/7314A-1 Communications Cards provide an alternative to using the RS-232-C serial port of the 7888 as the console device. When the RS-232-C port of the 7888 is not used as the console device, it can be used as a PC-compatible COM1 port. The COM1 port can then be used to interface to a modem, mouse, or serial printer.

When the 7350-01 EGA/Keyboard Card is installed in System 2 and connected to a PC/XT-compatible keyboard, it is automatically assigned by the STD ROM BIOS as the Console port. See Figures 3-1 and 3-2.

If a PC/XT-compatible keyboard is not connected to the 7350-01 EGA/Keyboard Card, it is not configured by the BIOS as the console interface. This allows you to run programs that display information on an EGA, CGA, or monochrome monitor through the BIOS (Video BIOS Int 10H) while using the console port on the 7888, 7314A-1, or 7312-1 to interface to a PC or terminal.

When a 7312-1 or 7314A-1 Communications Card is installed in your system (and the 7350-01 EGA/Keyboard Card is **not** installed), Channel A is automatically assigned by the STD ROM BIOS as the Console port. See Figure 3-3. If a 7312-1 or 7314A-1 communications card is installed in the system in addition to the 7350-01 EGA/Keyboard Card, Channel A is used as a general-purpose serial device, as are the rest of the channels on either of these cards. See Figure 3-1.

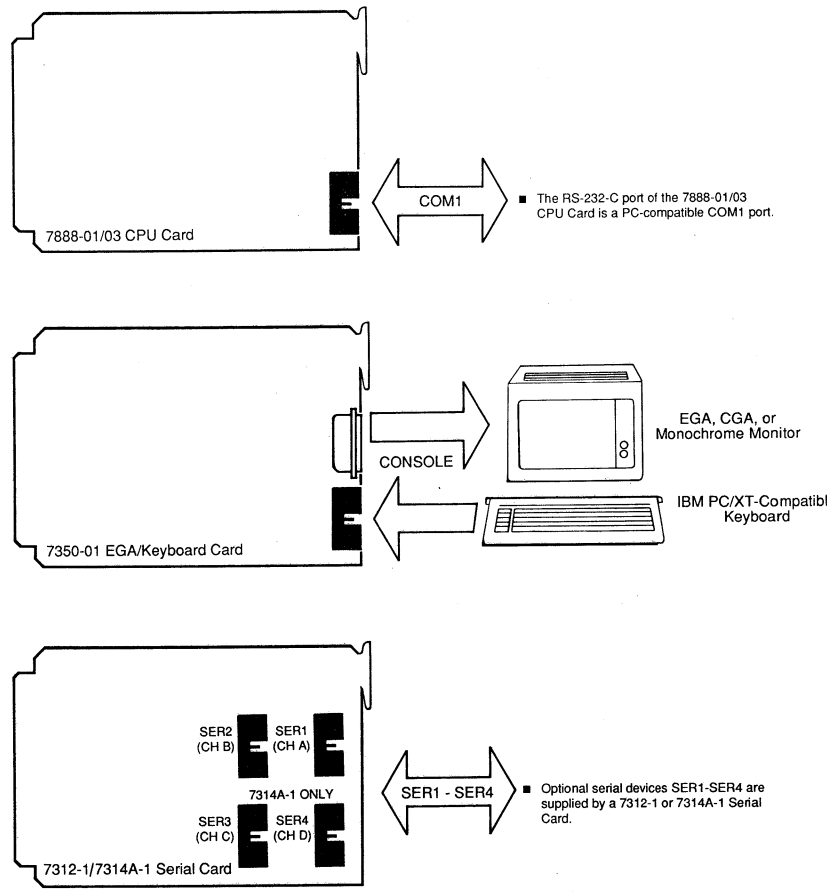

7888-01/03 CPU Card
COM1
The RS-232-C port of the 7888-01/03 CPU Card is a PC-compatible COM1 port.
7350-01 EGA/Keyboard Card
CONSOLE
EGA, CGA, or Monochrome Monitor
IBM PC/XT-Compatible Keyboard
7312-1/7314A-1 Serial Card
SER2 (CH B)
SER1 (CH A)
7314A-1 ONLY
SER3 (CH C)
SER4 (CH D)
SER1 - SER4
Optional serial devices SER1-SER4 are supplied by a 7312-1 or 7314A-1 Serial Card.

Figure 3-1. DOS Device Assignments When the 7350-01 and 7312-1/7314A-1 Cards Have Been Added to System 2

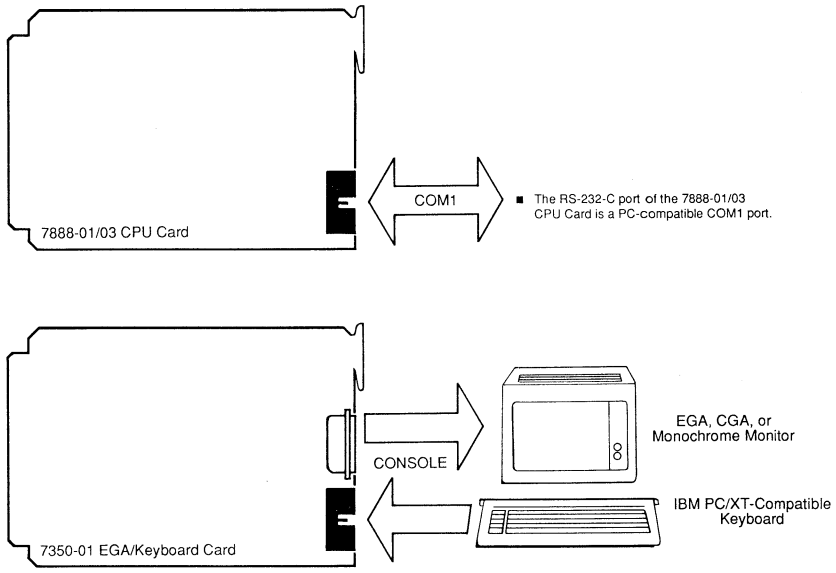

7888-01/03 CPU Card
COM1
■ The RS-232-C port of the 7888-01/03 CPU Card is a PC-compatible COM1 port.
7350-01 EGA/Keyboard Card
CONSOLE
EGA, CGA, or Monochrome Monitor
IBM PC/XT-Compatible Keyboard

Figure 3-2. DOS Device Assignments When the 7350-01 EGA/Keyboard Card Has Been Added to System 2

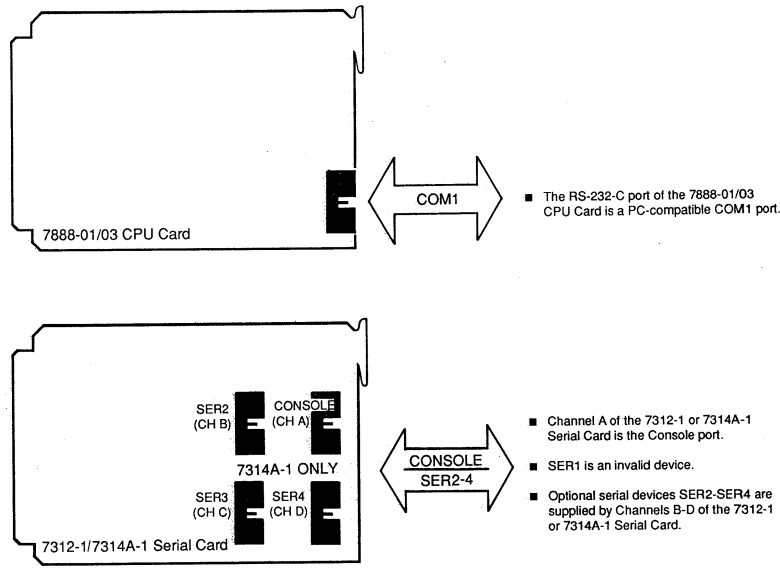

7888-01/03 CPU Card
COM1
The RS-232-C port of the 7888-01/03 CPU Card is a PC-compatible COM1 port.
SER2 (CH B)
CONSOLE (CH A)
7314A-1 ONLY
SER3 (CH C)
SER4 (CH D)
7312-1/7314A-1 Serial Card
CONSOLE
SER2-4
Channel A of the 7312-1 or 7314A-1 Serial Card is the Console port.
SER1 is an invalid device.
Optional serial devices SER2-SER4 are supplied by Channels B-D of the 7312-1 or 7314A-1 Serial Card.

Figure 3-3. DOS Device Assignments When the 7312-1 or 7314A-1 Serial Card Has Been Added to System 2

Chapter 6 provides additional information regarding DOS device assignments.

If neither of the Communications Cards (7312-1 or 7314A-1) nor the 7350-01 EGA/Keyboard Card is installed in the system, the RS-232-C port of the 7888 Processor Card is automatically assigned by the STD ROM BIOS as the Console port. Since COM1 is available only on the 7888 card and has been assigned as the Console port, COM1 is unavailable as a serial port. See Figure 3-4.

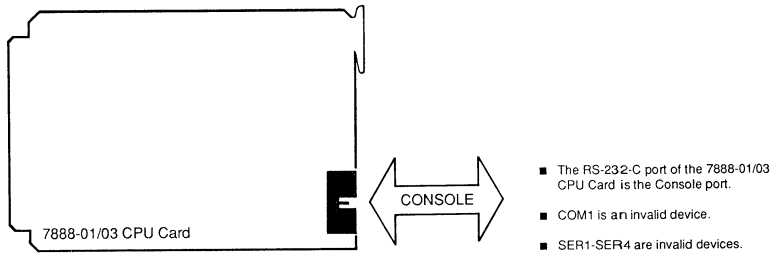

7888-01/03 CPU Card
CONSOLE
The RS-232-C port of the 7888-01/03 CPU Card is the Console port.
COM1 is an invalid device.
SER1-SER4 are invalid devices.

Figure 3-4. DOS Device Assignments of the Standard Configuration of System 2

3.2 INTERFACING THROUGH A MONITOR AND KEYBOARD

System 2 can be operated standalone as an industrial PC with the installation of the 7350-01 EGA/Keyboard Card, an IBM PC/XT compatible keyboard, and an EGA, CGA, or monochrome monitor. Since the hardware is PC-compatible, you can use off-the-shelf software packages that use graphics to display data. You can also generate custom graphics to display system status or information.

If your target system has the system memory required by the software development packages, application programs may be developed on System 2 using PC-compatible editors, assemblers, compilers, libraries, and debuggers that can be loaded into the system. The software can be loaded into the system through 3.5- or 5.25-inch floppy disk drive(s) or can be downloaded from the PC using the STD.EXE utility, dependent on your system configuration. Sections 3.5 through 3.7 provide information on how to connect an IBM PC/XT or PC/AT (or compatible) to System 2.

The STD.EXE utility is included on the STD-DOS Utilities Disk. Instructions for using the STD.EXE utility are provided at the end of this chapter. Copy-protected software cannot be downloaded from a PC and must be loaded into the system through floppy disk drives.

Most software development packages require a minimum of 256K bytes of system memory to operate. Refer to the data sheet of the specific software package for the amount of memory required to load and run the software. You can install the optional 7716-01 and 7716-02 256K-Byte CMOS Static RAM Memory Cards to expand system memory to 384K or 640K bytes.

3.2.1 Connecting a Monitor and Keyboard to SYSTEM 2

The *7350 User's Guide*, included with the card, provides details on the installation and operation of the EGA/Keyboard Card. Because you can use several types of monitors (EGA, CGA, or monochrome) with the 7350-01, there are some switch settings and a jumper that must be configured before connecting the card to a monitor. The two-wire Interrupt Interconnect cable should be connected between the 7888-01/03 CPU Card and the 7350-01 card.

If you are using a Hercules monitor, the HERC.EXE program should be run before Hercules-compatible applications software is run. The HERC.EXE utility program is included in the STDUTILS subdirectory of the STD-DOS Utilities disk.

If you did not order the 7350-01 EGA/Keyboard Card but would like more information, please refer to the 7350 Data Sheet contained in the *Systems Data Book*. The Pro-Log *Systems Data Book* is available free upon request and includes data sheets on System 2 accessory cards.

The remainder of this chapter describes how to connect the console port of your System 2 to a PC or terminal for system control. If you are using the 7350-01 EGA/Keyboard Card to interface to an EGA, CGA, or monochrome monitor and IBM PC/XT-compatible keyboard, skip to Chapter 4 for information on how to operate the system.

3.3 INTERFACING THROUGH A TERMINAL OR PC

You can use a terminal or PC as the operator console for your system. In this configuration, the terminal or PC is connected to the RS-232-C port of the 7888 CPU Card or Channel A of the optional 7312-1 or 7314A-1 Communications Card. If you are using the RS-232-C serial port of the 7888 CPU Card as the console port, you must connect the RC713 Null Modem with the RC716 Interconnect Cable to interface the system to the terminal or PC. The RC713 and RC716 are included with your system.

Because the RS-232-C serial port of the 7888 is configured as a PC-compatible COM1 port, the null modem is required to convert this port from a DTE (Data Terminal Equipment) configuration to a DCE (Data Communication Equipment) configuration. If you are using Channel A of the 7312-1 and 7314A-1 cards you do not need the RC713 null modem. Channel A of the 7312-1 and 7314A-1 cards is already configured as DCE and doesn't require the use of a null modem.

3.4 CONNECTING A TERMINAL TO SYSTEM 2

Step 1: Configure your terminal for the correct data communication parameters.

Configure the terminal for 19.2K baud, 8 data bits per character, 1 stop bit per character, and no parity. This is usually done via switches on the back of the terminal or through an initialization procedure from the keyboard. If you are changing switches at the back of the terminal while it is on, some terminals require that the system be turned off and on again before the new settings are active. Refer to your terminal's user's manual.

Step 2: Establish a physical interconnection between the terminal and your system.

If you are using the 7888 serial port as the Console port, follow the instructions outlined in step 2a. If you are using Channel A of the 7312-1 or 7314A-1 card as the console, follow the instructions outlined in step 2b.

Step 2a. If you are using the 7888 serial port as the Console port, attach the female connector of the null modem to the male 25-pin D-connector of the RC716 Interconnect Cable (or equivalent cable). Connect the male D-connector of this cable assembly (cable and null modem) to the serial port of the terminal.

Next, connect the transition connector at the other end of the cable assembly to J2 of the 7888 CPU Card (J2 is the 26-pin transition connector at the bottom of the card). The connectors on the RC716 cable and 7888 serial port are keyed to ensure proper connection. See Figure 3-5.

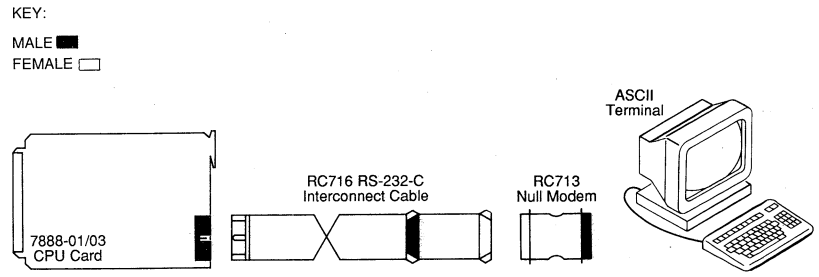


Figure 3-5. Connecting the RS-232-C Port of the 7888-01/03 CPU Card to an ASCII Terminal

Step 2b. If you are using Channel A of the 7312-1 or 7314A-1 card as the console, connect the 25-pin male D-connector of the RC716 cable (or equivalent cable) to the serial port of the terminal.

Next, connect the transition connector at the other end of the RC716 cable to J4 of the 7312-1 or 7314A-1 card (J4 is the 26-pin transition connector, closest to the card ejector at the top of the card). The connectors on the cable and serial cards are keyed to ensure proper connection.

Refer to the *7312/7314A User's Manual* for locations of the ports. Channel A has been preconfigured at the factory for proper operation as the console device and should not be changed unless you are using it as a general-purpose serial device. Refer to Chapter 6 for information on how the 7312-1 and 7314A-1 cards can be used to provide additional DOS serial devices in your system.

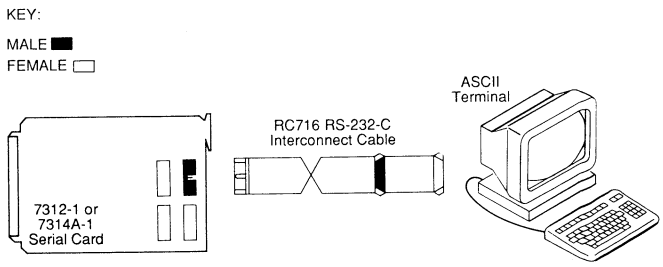


Figure 3-6. Connecting Channel A of the 7312-1 or 7314A-1 Serial Card to an ASCII Terminal

If you are using a terminal to interface to System 2, skip to Chapter 4 for information on how to operate the system.

3.5 USING A PC FOR SYSTEM CONTROL AND APPLICATION DEVELOPMENT

Connecting a PC to your system offers a powerful distributed processing architecture that allows the PC to tap into the control capabilities of the STD BUS. The PC could be used in much the same way as a terminal but, in addition, provides the ability to upload/download files and data.

If not required as a dedicated system controller, the PC can be used most of the time for office activities, such as word processing or new program generation, and periodically communicate with System 2 to load new programs or to gather data and tool status. The PC could then create management reports, spreadsheets, etc.

3.5.1 The PC as a Development System

The PC can be used to develop application programs for System 2. The programs can be written using any editor or word processor which runs in the PC. Because System 2 is PC-compatible, you can choose from a wide variety of PC-compatible languages. Pro-Log distributes the BRIEF Editor, sourced by UnderWare as well as Microsoft's Assembly, C, and BASIC languages.

After the program is assembled or compiled, it can be transferred to System 2 via floppy disk or downloaded to System 2 using the STD.EXE utility program, dependent on your system configuration. The STD.EXE program is provided on the STD-DOS Utilities Disk. Once the program is transferred to the system, it can be debugged using the debugger resident with MS-DOS. Alternatively, you can use a symbolic debugger, such as Microsoft's SYMDEB debugger. For more information on the development and execution of application programs, refer to Chapter 7.

3.5.2 PC Requirements

If the IBM PC, PC/XT, PC/AT, or hardware-compatible PC is used as a development system and/or system controller, it should have at least one floppy disk drive, 256K bytes of system RAM, and an IBM Asynchronous Adapter Card (or hardware compatible adapter card that uses the 8250 UART) to provide a COM1 port. Pro-Log's RC716 Interconnect Cable is used to interface the PC to System 2.

To interface the system to an IBM PC/AT, an IBM adapter cable is required. This adapter converts the 9-pin connector of the AT's asynchronous communication port to a 25-pin male D connector. The IBM part number for this adapter cable is 6450242.

3.6 CONNECTING THE PC TO SYSTEM 2

If you are using the 7888 serial port as the Console port, follow the instructions outlined in section 3.6.1. If you are using Channel A of the 7312-1 or 7314A-1 card as the Console port, follow the instructions in section 3.6.2.

3.6.1 Using the 7888's RS-232-C Port

When you are using the 7888 serial port as the Console port, attach the male D-connector of the RC713 Null Modem to the 25-pin female D-connector of the RC716 Interconnect Cable (or equivalent cable). Connect the female D-connector of this cable assembly to the PC's COM1 serial port located at the back of the PC.

Next, connect the transition connector at the other end of the RC716 cable to J2 of the 7888 card (J2 is the 26-pin transition connector at the bottom of the card). The connectors on the cable and CPU card are keyed to ensure proper connection.

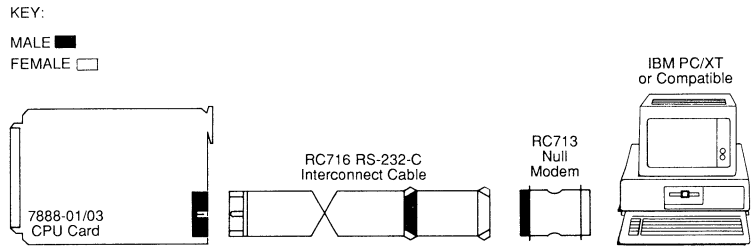


Figure 3-7. Connecting the RS-232-C Port of the 7888-01/03 CPU Card to COM1 of the IBM PC/XT or Compatible

When using an IBM PC/AT, an adapter cable (IBM part number 6450242) is required. The female D-connector of the cable assembly is connected to the 25-pin male D-connector of the IBM adapter cable.

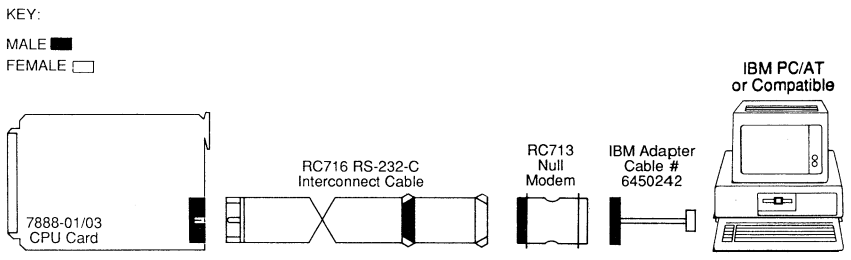


Figure 3-8. Connecting the RS-232-C Port of the 7888-01/03 CPU Card to COM1 of the IBM PC/AT or Compatible

3.6.2 Using the RS-232-C Port of the 7312-1 or 7314A-1

When using Channel A of the 7312-1 or 7314A-1 card as the console, connect the 25-pin female D-connector of the RC716 cable (or equivalent cable) to the PC's COM1 serial port located at the back of the PC.

Next, connect the transition connector at the other end of the RC716 cable to J4 of the 7312-1 or 7314A-1 card (J4 is the 26-pin transition connector, closest to the card ejector, at the top of the card). Refer to the *7312/7314A User's Manual* for locations of the ports. The connector on the cable and communications cards is keyed to ensure proper connection.

Channel A has been preconfigured at the factory for proper operation as the console device and should not be changed unless you are reconfiguring the channel to be used as a general-purpose serial device. Refer to Chapter 6 for information on how the 7312-1 and 7314A-1 cards can be used to provide additional DOS serial devices in your system.

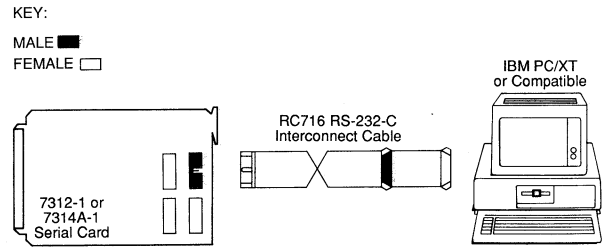


Figure 3-9. Connecting Channel A of the 7312-1 or 7314A-1 Serial Card to COM1 of the IBM PC/XT or Compatible

When using an IBM PC/AT, an adapter cable (IBM part number 6450242) is required. The female D-connector of the cable assembly is connected to the 25-pin male D-connector of the IBM adapter cable.

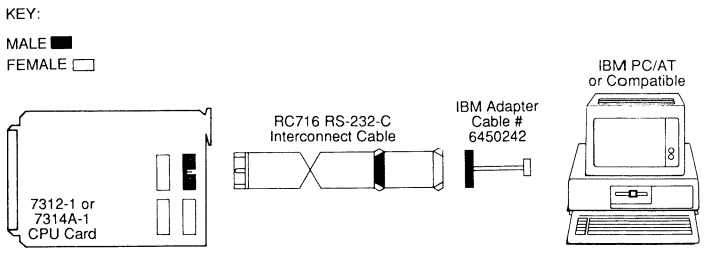


Figure 3-10. Connecting Channel A of the 7312-1 or 7314A-1 Serial Card to COM1 of the IBM PC/AT or Compatible

3.7 THE STD.EXE TERMINAL EMULATION UTILITY

STD-DOS sends an ANSI escape sequence " ESC [2 J" to the console to clear the display. The PC must have the ANSI.SYS device driver installed in the CONFIG.SYS file to recognize the ANSI escape sequence. The ANSI.SYS device driver is included with the PC-DOS system disk. The following command can be used to create a CONFIG.SYS file that includes ANSI.SYS.

```
COPY CON CONFIG.SYS
DEVICE=ANSI.SYS
<Ctrl Z>
```

The ANSI.SYS file must be on the PC system disk during boot for the CONFIG.SYS file to find it. Do not use the ANSI.SYS device drive in the CONFIG.SYS file in System 2 if you are using the STD.EXE utility. The two programs are not compatible and will lock up System 2.

The STD.EXE utility program is included on the STD-DOS Utility Disk.

1. Insert the disk in your PC. For this example we will assume you have loaded the disk in Drive A.

2. Start the STD utility:

   A>STD

   Even with System 2 disconnected or powered down, the PC monitor should display:

   STD version 2.XX
   (c) Copyright 1985, 1986 Pro-Log Corp.

3. When System 2 is on and the operating system is booted, the PC displays the STD prompt:

   STD A:\>

NOTE: If your system boots from a hard disk, the system prompt will be:

   STD C:\>

The "STD C:\>" means the System 2 now has control of the display. The following function keys are active:

KEY	FUNCTION
Alt-F1	Displays Help Screen
Alt-F2	Exit the STD utility
Alt-F3	Copy PC file(s) to System 2
Alt-F4	Copy System 2 file(s) to PC
Alt-F5	Reset System 2

Pressing <Alt> <F1> displays the help screen. The help screen displays the available operations and the keys used to execute them. To return from the help screen, press any key.

Pressing <Alt> <F2> exits the STD.EXE program and returns control to the PC operating system.

Pressing <Alt> <F3> begins the download utility. The following example transfers filename.com from the default drive in the PC to Drive B in System 2. Underlined characters are entered by you.

```
STD A:\>  Copy from PC to STD BUS --
          PC filename: filename.com
      STD directory: b:
          Copying file(s)...
      filename.com
          1 file(s) copied
```

The PC filename and STD directory can be specified using a pathname. Wildcard characters can be used when specifying the PC filename. Wildcard characters are explained in the *DOS Reference* manual.

Pressing <Alt> <F4> begins the upload utility. The following example transfers filename.com from Drive B in System 2 to Drive B in the PC.

```
STD A:\>    Copy from STD BUS to PC --
            STD filename: b:filename.com
        PC directory: b:
            Copying file(s)...
        filename.com
            1 file(s) copied
```

The STD filename and PC directory can be specified using a pathname. Wildcard characters can be used when specifying the STD filename.

Pressing <Alt> <F5> resets System 2. This is similar to pressing <Ctrl> <Alt> <Del> on the PC.

3.7.1 Using COM2

STD.EXE is written to communicate to System 2 through COM1 of the PC. If the COM1 port of your PC is connected to another device (such as a serial printer, serial mouse, or modem), the STD.EXE program can be operated through the COM2 port of the PC. Before executing the STD utility, enter the following command:

SET STD=COM2

This command can be entered before executing the STD utility or included in the PC AUTOEXEC.BAT file.

3.7.2 Printing the Screen

The Print Screen function of the PC can be used to print the information currently on the screen while running the STD.EXE program. Output is sent to the PC's printer, not the PRN device of System 2. The Printer Echo function, executed by pressing the <Ctrl> and <Esc> keys simultaneously, is used to send screen output to the PRN device of System 2. It will also print to the PRN device of the PC. Refer to the *DOS Reference* manual for more information.

Chapter 4
System Operation

Now that you have connected a monitor/keyboard, terminal, or PC to the system, you are ready to power-on the system and begin system operations. System 2 operates much the same as an IBM PC/XT. System 2 can be booted from floppy or hard disk drives, just like a personal computer, but it can also operate with semiconductor disk drives.

System 2 tries to boot the operating system from the disk drives in the following order:

1. Floppy drive A
2. Hard disk drive C
3. ROM disk drive A
4. Optional semiconductor disk drives H-U

If you do not want to boot from the factory-installed hard disk, press the <Esc> key during the system RAM test that is performed at power-up. If an operating system disk has not been inserted into floppy drive A, the system will try booting from the semiconductor disks starting with ROM disk A.

If the operating system cannot be loaded from any of these disks, a failure message is displayed on the console and the system is halted.

4.1 BOOTING SYSTEM 2 FROM FLOPPY DISK

System 2 can support up to two 3.5-inch or 5.25-inch floppy drives via the 7391-01 Floppy Disk Controller Card. If installed, the floppy drives are accessed as drives A and B. If a floppy disk with system files has been inserted into floppy drive A, the system boots from this disk.

System 2 Model 20 includes an internal 3.5-inch floppy disk drive A and a 3.5-inch MS-DOS 3.2 Operating System Disk. Model 20 can be booted by inserting the operating system disk (label faces to the right) in drive A immediately after applying power to the system.

If the system does not boot after a minute or so, press the <Ctrl>, <Alt>, <Del> keys at the same time if you're using a PC or PC monitor and keyboard. If you've connected the system to a terminal, the system can be restarted by pressing the <Esc> <c> keys simultaneously.

When booting the system from a floppy disk, the following should be displayed on the terminal or monitor:

```
STD-DOS version 2.XX
(C) Copyright 1985, 1986, 1987 Pro-Log Corp.

XXX Kbytes of System RAM

Microsoft(R) MS-DOS(R) Version 3.20
            (C)Copyright Microsoft Corp 1981-1986

A>PROMPT STD $p$g

STD A:\>path=a:\;a:\msutils;a:\stdutils;

STD A:\>
```

4.2 BOOTING SYSTEM 2 FROM HARD DISK

If you ordered Model 10 with Option 021 or Model 20 with Option 024, an internal 3.5-inch hard (Winchester) disk drive has been factory-installed in your system. The system files have been copied to the hard disk. If you do not have floppy drives in your system or you have not loaded a system disk in floppy drive A, the system boots from hard disk immediately after power has been applied.

When booting the system from the hard disk, the following should be displayed on the terminal or monitor:

```
STD-DOS version 2.XX
(C) Copyright 1985, 1986, 1987 Pro-Log Corp.

System Memory Test in Progress ...
XXX Kbytes of System RAM have been tested

Microsoft(R) MS-DOS(R) Version 3.20
(C)Copyright Microsoft Corp 1981-1986

C>prompt STD $p$g

STD C:\>path=c:\;c:\msutils;c:\stdutils;a:\;

STD C:\>
STD C:\>
```

4.3 BOOTING SYSTEM 2 FROM ROM DISK

If the operating system has not been loaded from floppy or hard disk, System 2 tries booting the operating system from ROM disk drive A, then from optional semiconductor disk drive H.

The standard configuration of Model 10 includes two ROM disk drives: A and B. ROM disk drive A contains the MS-DOS 3.2 operating system, while drive B is a 256K-byte user-supplied ROM disk. If you order Model 10 with Option 020, a 128K-byte battery-backed RAM disk is provided instead of user-supplied ROM disk B.

When booting Model 10 from ROM disk A, the following should be displayed on the terminal or monitor:

```
STD-DOS version 2.XX
(C) Copyright 1985, 1986, 1987 Pro-Log Corp.

XXX Kbytes of System RAM

Bad or missing B:\DEVICE1.SYS

Bad or missing B:\DEVICE2.SYS

Microsoft(R) MS-DOS(R) Version 3.20
(C)Copyright Microsoft Corp 1981-1986

A>prompt STD $p$g

STD A:\>path=a:\;

STD A:\>b:\startup
Bad command or file name

STD A:\>
```

The CONFIG.SYS file on Drive A allows 2 user-written device drivers to be installed on ROM Disk Drive B (or the optional battery-backed RAM Disk drive B). Installable device drivers can be added to System 2 using the device command. The device command must be included in the CONFIG.SYS file on Drive A so that DOS loads the driver during startup.

When the system boots, the messages "Bad or missing B:\DEVICE#.SYS" indicates only that DOS did not find that specific installable device driver on Drive B.

The AUTOEXEC.BAT file on ROM Disk A searches Drive B for a file called "STARTUP" during power-on. If you do not have a file called STARTUP.BAT, STARTUP.EXE, or STARTUP.COM on Drive B, the system returns the message "Bad command or file name." This startup file allows System 2 to run programs at power-up.

The system is functioning properly if the prompt "STD A:\>" is displayed on the last line.

Assuming that the system is operating correctly, you can now execute MS-DOS commands or begin development of your application programs.

If you have ordered Model 10 with Option 025 or 026, the two semiconductor disks are configured as drives H and I instead of drives A and B. This configuration allows you to integrate external floppy disk drives that can only be accessed as drives A and B.

When booting Model 10 from ROM disk H, the following should be displayed on the terminal or monitor:

```
STD-DOS version 2.XX
(C)Copyright 1985, 1986, 1987 Pro-Log Corp.

XXX Kbytes of System RAM

Bad or missing I:\DEVICE1.SYS

Bad or missing I:\DEVICE2.SYS

Microsoft(R) MS-DOS(R) Version 3.20
(C)Copyright Microsoft Corp 1981-1986

H>prompt STD $p$g

STD H:\>path=h:\;

STD H:\>i:\startup
Bad command or file name

STD H:\>
```

4.4 IN CASE OF DIFFICULTY

If the system doesn't start up correctly, check the RS-232-C connection between the terminal or PC and System 2. If you are using a terminal as the operator console, check the settings of the terminal (some require that they be powered off and on to reset the baud rate and other communication parameters). If you're connecting the RS-232 port on the 7888 CPU card to a terminal or PC, check to ensure the RC713 Null Modem is also connected.

If the 7350-01 EGA Card is installed in the system, make sure that the cables connecting the monitor and keyboard are secure. Also make sure that the two-wire connector between the 7888-01 and the 7350-01 is in place. Refer to the *7350 EGA/Keyboard User's Guide* for detailed installation instructions.

Next, ensure that the system is receiving power. The LED on the 7909A Terminator Card should be lit to indicate that the +5V power supply is available on the STD BUS backplane. The ±12V supplies can be checked at the back of the card rack on the motherboard. Refer to Figure 10-7, page 10-12, for the locations of the voltage and ground points available at the terminal strips on the bottom and top of the motherboard.

If all voltages are present on the bus, check to make sure that the LED on the 7713 System Disk Card or the LED on the floppy disk drive (Model 20) blinks when System 2 is powered on. If everything you've checked seems to be working correctly but the system is still not communicating with the terminal or PC, power everything down and repeat the steps outlined in Chapter 3.

If the system is still not operational, please call us at (800) 538-9570 and ask for System 2 application support.

4.5 PREPARING TO MOVE SYSTEM 2

BEFORE MOVING OR SHIPPING a System 2 that has a hard disk installed, you should first park the heads.

The PARK.EXE utility prepares the System 2 hard disk drive for transport. The PARK.EXE utility is included in the STDUTILS subdirectory on the hard disk and the STD-DOS Utilities disk. It is also provided on the operating system floppy or ROM disk.

When preparing to move the system, "park c:" should always be the last command executed on a System 2 that includes a hard disk. Before executing this utility, the system default drive must be changed to a drive other than C (that is, the default must not be the hard disk itself).

The head mechanism of the hard disk is correctly parked by doing the following:

1. Change the system default drive to a drive other than drive C.

   Example:

   ```
   STD C:\>A: <CR>

   STD A:\>
   ```

2. Type "park c:" followed by a carriage return (enter key).

   ```
   STD A:\>park c: <CR>
   C Drive Parked....Ready to move
   STD A:\>
   ```

The head mechanism is now parked and the system can be turned off and moved.

Chapter 5 shows how to install additional user-installed accessories to the system.

Chapter 5

Installing Additional System Components

5.1 INTRODUCTION

Other STD BUS cards installed by the factory or optional cards that you install yourself can be placed anywhere in the card rack. However, it is recommended that the 7909A terminator card be placed at the opposite end of the motherboard from the 7888 CPU card. Any STD BUS cards that use the priority chain (PCI and PCO) should be grouped together with no empty slots between them. Refer to the card's data sheet or user's manual for priority chain information.

The memory, I/O, and interrupt map of the IBM PC, PC/XT, and PC/AT have been adhered to as closely as possible in System 2. Many of the peripheral devices used in System 2 are compatible with the PC. Compatible memory, I/O, and interrupt mapping allow System 2 to run application software written for the PC.

It is also important to consider the power requirements of your system when adding other STD BUS cards or other system components. The power requirements of System 2, Model 10 and 20, are provided in the electrical specifications in Chapter 12.

Cards can be easily removed or installed by loosening the two thumb screws that hold the retaining bar in place. To add a card to your system, slide the card in the motherboard using the guides installed on the top and bottom of the card rack. You should feel the card snap into the connector of the motherboard when it has been inserted correctly.

When removing a card, lift the retaining bar so that you can lift the ejector tab on the card. When the ejector tab is lifted, it will pull the card away from the connector of the motherboard.

5.2 SYSTEM 2 SUPPORTS UP TO 640K BYTES OF SYSTEM MEMORY

MS-DOS uses system memory to load and execute DOS commands and your application programs. MS-DOS 3.2 occupies approximately 32K bytes of system memory, with the remainder of system RAM available for loading and executing your programs and for data buffering. Figure 5-1 illustrates the System 2 memory map.

64K	FFFFF F0000	ROM BIOS (7888)	
64K	EFFFF E0000	ED000-EDFFF SCSI BUFFER EA000-EBFFF FLOPPY BUFFER E0000-E01FF MEMORY DISK	Pro-Log Reserved
128K	DFFFF C0000	DC000-DFFFF D0000-D7FFF P-L RESERVED CC000-CEFFF FLOPPY ROM C8000-CBFFF SCSI ROM C0000-C7FFF EGA ROM	ROM BIOS Extension
128K	BFFFF A0000	B8000-BFFFF CGA/EGA B0000-B3FFF MONOCHROME	Graphics Memory Display
512K	9FFFF 20000	(ON STD BUS)	System RAM
128K	1FFFF 00000	(ON 7888)	

Figure 5-1. System 2 Memory Map

System 2 can address up to 640K bytes of system RAM, mapped as a contiguous block starting at address 00000 hex. System 2 is shipped with at least 128K bytes of system RAM. This RAM resides on the 7888 CPU Card and is mapped at addresses 00000 to 1FFFF hex. Additional 7716-01 and 7716-02 CMOS static RAM cards can be installed to expand system memory to 384K bytes or 640K bytes. You can have these memory cards installed when you initially purchase your system or add them later when your requirements change.

The 7716-01 is mapped to expand system memory from 128K bytes to 384K bytes. The 7716-02 is mapped to expand memory from 384K bytes to 640K bytes. The 7716-01 must be installed if the 7716-02 is installed. CMOS static RAMs are used on the 7716 cards to minimize power requirements, extend the operating temperature range, and avoid wait states.

5.3 CUSTOMIZING YOUR SYSTEM I/O

We offer a complete line of peripheral, analog, digital, and industrial I/O cards. These cards provide data communication channels and interfaces for monitoring and controlling real-world devices. In most cases, software support for these I/O cards is provided by DOS device drivers standard with System 2 or through I/O drivers routines available in the optional STD-LIB Software Library.

5.3.1 I/O Options

A list of I/O cards and software support available from Pro-Log is provided in Figure 5-2. The Pro-Log *Systems Data Book*, available free upon request, includes data sheets on these accessory cards.

MODEL	DESCRIPTION (Software Support)
Peripherals:	
7303	Keyboard/Display card (STD-LIB Driver)
7312-1	Dual serial card (DOS Device Driver)
7312-03	Dual serial card (STD-LIB Driver)
7314A-1	Quad serial card (DOS Device Driver)
7314A-03	Quad serial card (STD-LIB Driver)
7350-01	EGA/Keyboard card & cables (BIOS extension)
7391-01	Floppy Disk Controller (BIOS extension)
7392-01	Printer controller card & cable (DOS Device Driver)
GPIB-961P	IEEE-488 card (DOS Applications Driver)

Figure 5-2. I/O Cards Available

Analog I/O:

RTI-1225	Analog I/O card (STD-LIB Driver)
RTI-1226	Analog Input card (STD-LIB Driver)
RTI-1260	Analog Input card (STD-LIB Driver)
RTI-1262	Analog Output card (STD-LIB Driver)
7401A-50	50 Hz Thermocouple Input card (Example routines)
7401A-60	60 Hz Thermocouple Input card (Example routines)

Industrial I/O:

7501	Medium Power DC Driver card (STD-LIB Driver)
7502	SPST Relay Output card (STD-LIB Driver)
7503	Optoisolated Input card (STD-LIB Driver)
7504	TRIAC Output card (STD-LIB Driver)
7506	Optoisolated Input card (STD-LIB Driver)
7507	General Purpose 24 I/O card (STD-LIB Driver)
7508	General Purpose 48 I/O card (STD-LIB Driver)
SO-102	Opto-22 Prototyping Kit (STD-LIB Driver)
SO-103	Opto-22 OPTOMUX Proto. Kit (STD-LIB Driver)

Digital I/O:

7605-0	Programmable 32-line I/O card (STD-LIB Driver)
7605-1	Programmable 32-line I/O card (STD-LIB Driver)
7614	64-line I/O card (STD-LIB Driver)

Utility:

7902	Utility DIP card
7903	General Utility card
7914	Decoded I/O Utility card (STD-LIB Driver)

Figure 5-2. I/O Cards Available (Continued)

5.3.2. I/O Mapping

The I/O map for System 2 is included in Figure 5-3. It is recommended that you do not map optional I/O cards in I/O addresses reserved for System 2 resources or areas reserved for Pro-Log or IBM future expansion. I/O cards not used to provide DOS devices should be mapped at I/O addresses 100-1EF hex.

8-BIT I/O CARDS

Any I/O operations that address 100 to 1FF hex drive IOEXP low on the STD BUS. All 8-bit addressable I/O cards should respond to IOEXP being low to prevent contention with cards addressed by 10 or 16 bits. When an 8-bit I/O card is hardware mapped at address

00-EF and configured to respond to IOEXP low, it is accessed at 100-1EF hex.

PORTS	USAGE
000-01F	*DMA controller #1(8237A)
020-03F	***Interrupt controller #1 (82C59A)**
040-05F	***Counter/timer (82C54)**
060-06F	***PPI (8255A-5, selected functions only)**
070-07F	*Real-time clock, AT-compatible
080-09F	*DMA page registers
0A0-**0AF**	**NMI port**
0B0-0BF	*Interrupt controller #2
0C0-0DF	*DMA controller #2
0E0-0EF	IBM RESERVED
0F0-0FF	*Math Coprocessor
100-1EF	Customer use
1F0-1FF	*Fixed disk (AT)
200-20F	*Game control
210-21F	*Expansion unit
220-24F	*IBM RESERVED
250-26F	PRO-LOG RESERVED
270-27F	***Printer #2 (7392 printer option)**
280-29F	**Memory disks (7714/15 memory disk options)**
2A0-2BF	PRO-LOG RESERVED
2C0-2CF	**SCSI card (7390 SCSI option)**
2D0-2DF	**Floppy controller card (7391 FDC option)**
2E0-2EF	PRO-LOG RESERVED
2F0-2FF	IBM RESERVED & Async comm #2
300-31F	Prototype cards
320-32F	*Fixed disk (XT)
330-35F	PRO-LOG RESERVED
360-36F	*IBM RESERVED
370-37F	***Printer #1 (7392 printer option)**
380-38F	*SDLC/BI-SYNC comm #2
390-39F	Customer use
3A0-3AF	*BI-SYNC comm #2
3B0-3BF	***Monochrome display/printer (7350/7392)**
3C0-3CF	*IBM RESERVED
3D0-3DF	***Color graphics display (7350 EGA option)**
3E0-3EF	*IBM RESERVED
3F0-3FF	*Diskette and **Async comm #1**
400-FFFF	(not used)

*IBM-compatible hardware
Boldface indicates I/O ports used in System 2

Figure 5-3. System 2 I/O Map

5.3.3 DOS Devices

For information regarding I/O facilities accessible through the operating system, refer to the information on DOS devices included in Chapter 6.

5.3.4 Additional I/O Facilities and Software Drivers

The STD-LIB software library (optional) provides I/O driver routines for over 22 digital and analog I/O cards. It also contains timing and memory diagnostic routines. All STD-LIB routines are written in assembly language for fast execution. They can be linked with programs written in Microsoft C, Microsoft Compiled BASIC, and Microsoft's Macro Assembler (all available from Pro-Log). STD-LIB version 2.1 and above is shipped on 3.5-inch and 5.25-inch floppy disks.

5.4 INTERRUPTS

Application programs written to run on System 2 can use interrupts 60-67 hex and C0-FF hex without conflicting with interrupts used for system operation. The interrupt map for System 2 is provided in Figure 5-4.

INTERRUPT	USAGE
00	**Divide by 0** (dedicated hdw interrupt)
01	**Single step** (dedicated hdw interrupt)
02	**NMI** (dedicated hdw interrupt)
03	**Breakpoint** (dedicated hdw interrupt)
04	**Overflow** (dedicated hdw interrupt)
05	***Print screen**
06-07	IBM RESERVED
08	***System timer hardware interrupt**
09	***Keyboard hardware interrupt** (7350 EGA option)
0A	***CRT vertical retrace** (7350 EGA option)
0B	**Customer hardware interrupt** (front of 7888)
0C	***Communications hardware interrupt**
0D	**Real-time clock hardware interrupt**
0E	**VBAT low hardware interrupt**
0F	**STD BUS INTRQ**
10	***Video display BIOS service** (7350 EGA option)
11	***Equipment check BIOS service**
12	***Memory size BIOS service**
13	***Diskette BIOS service** (7391 FDC option)
14	***Communications BIOS service**
15	*Cassette
16	***Keyboard BIOS service** (7350 EGA option)
17	***Printer BIOS service** (7392 printer option)
18	Resident BASIC
19	***Bootstrap**
1A	***Time and date BIOS service**
1B	***Keyboard break**
1C	***Timer tick**
1D	***Video initialization** (7350 EGA option)
1E	***Diskette parameters** (7391 FDC option)
1F	***Video graphics** (7350 EGA option)
20-3F	**MS-DOS RESERVED**
40-5F	IBM RESERVED
60-67	Customer use
68-70	PRO-LOG RESERVED
71	**SERn software interrupt**
72	PRO-LOG RESERVED
73	**Clock software interrupt**
74-76	PRO-LOG RESERVED
77	**Memdisk I/O BIOS service**
78	**Mechanical drive software int** (7390/7391)
79	**SCSI software interrupt** (7390 SCSI option)
7A-7F	PRO-LOG RESERVED
80-9F	**SERn hardware interrupts** (7312/14 option)
A0-A3	PRO-LOG RESERVED
A4	**SCSI hardware interrupt** (7390 SCSI option)
A5	**Floppy hardware interrupt** (7391 FDC option)
A6-BF	PRO-LOG RESERVED
C0-FF	Customer use

*IBM-compatible interrupt routine
Boldface indicates interrupts used in System 2

Figure 5-4. System 2 Interrupt Map

Chapter 6
DOS Devices

6.1 CHARACTER DEVICES

Character devices perform serial character I/O operations. Examples of character devices are CON, COM1, AUX, PRN, etc. Figure 6-1 lists the available DOS character devices and their default parameters.

6.1.1 CON (Console)

The Console device is both an input (keyboard) and output (display) device. It is the MS-DOS standard input and output device for operator interface. The standard configuration of System 2 is shipped with the RS-232-C port on the 7888 CPU card configured as the Console. Alternatively, you can use the 7312-1 or 7314A-1 Communications card or the 7350-01 EGA/Keyboard Card to provide the console port.

THE RS-232-C PORT ON THE 7888 AS THE CONSOLE DEVICE

If a 7312-1 or 7314A-1 Communications card or 7350-01 EGA/Keyboard Card is **not** installed in the system, the RS-232-C port of the 7888 CPU Card is assigned as the Console port. If a 7350-01, 7312-1, or 7314A-1 card has been installed in the system, the RS-232-C port on the 7888 CPU card is dedicated as the COM1 port. Whether the RS-232-C port of the 7888 is used as the Console (CON) or as the COM1 device, this port is configured as Data Terminal Equipment (DTE) and requires the use of a null modem when connected to a PC or terminal.

USING THE 7350 EGA CARD AS THE CONSOLE DEVICE

The STD ROM BIOS first checks to see if a 7350-01 EGA Card with a PC/XT compatible keyboard is installed in the system. If installed, it is assigned as the Console. Refer to the *7350 EGA/Keyboard Card User's Guide* for information regarding the installation and operation of the 7350-01.

CHARACTER DEVICE	PHYSICAL LOCATION	INTERFACE	PROTOCOL	BAUD RATE	DATA BITS	PARITY	STOP BITS
CONsole	7888-01/03[1] 7312-1 or 7314A-1[2] 7350-01	RS-232-C (DTE) RS-232-C (DCE) EGA, CGA, or Monochrome Video and PC/XT Keyboard	Asynch. Full Duplex Asynch. Full Duplex -	19.2K 19.2K -	8 8 -	None None -	1 1
COM1	7888-01/03[3]	RS-232-C (DTE)	Asynch. Full Duplex	2400	8	None	1
SER1	7312-01 or 7314A-01[4]	RS-232-C (DCE)	Asynch. Full Duplex	9600	8	None	1
SER2 (PRN)[5]	7312-01 or 7314A-1	RS-232-C (DCE)	Asynch. Full Duplex	2400	8	None	1
SER3	7314A-1	RS-232-C (DCE)	Asynch. Full Duplex	9600	8	None	1
SER4	7314A-1	RS-232-C (DCE)	Asynch. Full Duplex	9600	8	None	1
LPT1 (PRN)[5]	7392-01	Parallel	Centronics	-	-	-	-
LPT2-LPT3	7392-01 (Remapped)	Parallel	Centronics	-	-	-	-
@CLOCK	7888-01/03	-	-	-	-	-	-
NUL	-	-	-	-	-	-	-

1. If 7312-1 or 7314A-1 or 7350-01 **is not** installed.
2. If 7350-01 **is not** installed.
3. If 7312-01, 7314A-01, or 7350-01 **is** installed.
4. If 7350-01 **is** installed.
5. If the 7392-01 Parallel Printer Card is installed, input and output to the standard print device (PRN) is directed to LPT1. If the 7312-1 or 7314A-1 Serial Card is installed and the 7392-01 **is not** installed, PRN is directed to SER2.

Figure 6-1. DOS Character Devices

USING A 7312-1 OR 7314A-1 COMMUNICATIONS CARD AS THE CONSOLE DEVICE

If the 7350-01 and keyboard are not installed in the system at power-up, the BIOS checks to see if a 7312-1 or 7314A-1 Communications Card is installed. If either of these optional I/O cards has been installed, the first port (Channel A) of the communications card is dedicated as the console. Note that only one of the 7312-1 or 7314A-1 cards can be installed in a system since the cards are I/O mapped at the same address.

USING HANDSHAKING ON THE CONSOLE PORT

If a 7312-1, 7314A-1, or 7888 provides the Console, the RS-232-C handshake lines can be used when transmitting data. If the 7312-1, 7314A-1, or 7888 console port is connected to a PC, all handshake lines must be connected.

If a terminal is connected to the console port, a simple two-wire interface without handshaking can be used. However, if the DTR line is connected and active, the RTS line from the terminal is checked and must be connected and active before data is sent from the terminal.

Refer to Chapter 3, *Selecting an Operator Interface*, for information on how to use a PC or terminal as an operator interface. Refer to the *7350 User's Guide* if you're using the 7350-01 EGA/Keyboard Card to connect the system to a monitor and keyboard.

6.1.2 COM1

If the RS-232-C serial port on the 7888 CPU Card is not used as the Console device, the BIOS automatically configures the port as the COM1 device. In this configuration, the 7888's RS-232-C port is IBM PC/XT-compatible. At initialization, output and input from the standard auxiliary device (AUX) is directed to COM1. However, you can use the MODE command to redirect I/O from the AUX port to another DOS device. You can use the CTTY command to redirect the standard input and output to COM1.

6.1.3 SER1, SER2, SER3, and SER4

The other four character devices (SER1, SER2, SER3, and SER4) provide support for the RS-232-C serial ports located on the optional 7312-1 or 7314A-1 Communications Cards. The 7312-1 provides two RS-232-C serial ports configured as SER1 and SER2, whereas the 7314A-1 provides four RS-232-C serial ports configured to support SER1 through SER4. However, if either of these cards is

installed and the 7350-01 EGA Card is **not** installed, Channel A is dedicated as the Console (CON) instead of SER1. In this case, all accesses to the device SER1 return an error.

Output and input from the standard print device (PRN) is directed to SER2, when a 7392-01 printer card in not installed, so that a serial printer can be attached to the system. PRN can also be redirected using the MODE command. The standard input and output can be redirected to any of the SER ports using the CTTY command.

Each of the serial ports on the 7312-1 or 7314A-1 card(s) can be configured as DCE (Data Communications Equipment) or DTE (Data Terminal Equipment) as required by the application. When Channel A of either of these cards is used as the console device, it should be configured as DCE, which is the as-shipped configuration. Refer to the *7312/7314A User's Manual*, shipped with the card, for information on the as-shipped configuration and how to reconfigure each port as DCE or DTE. To ensure that the card operates properly in the system, no other changes should be made to the as-shipped configuration.

All ports configured as SER devices (SER1, SER2, SER3, and SER4) can use the RS-232-C handshake lines when transmitting data. If DTR from the terminal is active, the RTS line from the terminal is checked and must also be active before data is sent from the terminal. If a simple two-wire interface without handshaking is required, the DTR line can be left unconnected.

Input to the SER ports is interrupt driven and a separate buffer is maintained for each SER device. In addition, when its input buffer is more than three-quarters full, the SER port sets its outgoing CTS line (if configured as DCE) inactive. When characters are read from the buffer and it becomes less than three-quarters full, CTS is set active again.

ADDITIONAL SERIAL PORTS

If your target system requires more RS-232-C serial ports than those available as DOS devices, the 7312-03 and 7314A-03 Communications Cards can be installed in the system. The STD-LIB software library (all versions) provides I/O driver routines to support the RS-232-C ports on these cards. Up to 32 RS-232-C ports can be supported in the target system, in addition to the four serial ports assigned as SER1 through SER4. The 7312-03 and 7314A-03 cards have been mapped to a starting I/O address of 160 to 167 hex. If more than one of these cards are used to provide additional serial ports, they should be mapped contiguously starting

at the default address. The user's manual for these cards provides details on I/O mapping and information regarding the configuration of each channel for DTE or DCE applications.

6.1.4 LPT1, LPT2, LPT3

If the 7392-01 Parallel Printer Card is installed, input and output to the standard print device (PRN) is directed to LPT1. If the 7392-01 is not installed, but a 7312-1 or 7314A-1 card is, PRN is directed to SER2. A jumper group on the 7392-01 configures the card as LPT1, LPT2, or LPT3. Therefore, up to three 7392-01 cards can be installed in a system.

6.1.5 @Clock

DOS uses the Clock device for marking file control blocks and directory entries with date and time as well as providing date/time services to applications programs. The user can enter a new time or date and can read the current time or date at any time from the console, using the DOS commands TIME and DATE.

Applications programs can use the DOS function calls 2A-2DH to get or set the time or date. The optional STD-LIB software library (version 2.0 or later) also provides timing routines that allow the application program to read or write to the clock, wait a specified period of time, and set future time marks that may be checked later to see if the time has expired. The Clock device is initialized by a hardware clock upon power-on. Setting the time through the TIME and DATE commands only changes the time in software, not in the hardware clock; therefore, any changes made will be lost on power down. To actually set the hardware clock, execute the SETRTC.BAT utility found on ROM Disk Drive A (or ROM Disk H), or on the MS-DOS 3.2 System disk. The hardware battery-backed time-of-day clock retains the time you have set for a minimum of 10 years.

6.1.6 NUL

The NUL (null) device is a nonexistent or dummy device used in application development and testing applications. Whenever input is read from this device, an immediate end-of-file is generated. When data is output to this device, the write operation is simulated, but no data is actually written.

6.1.7 Modifying or Redirecting Character I/O

Many default parameters of the serial and printer devices can be modified using the MODE and SERMODE commands. MODE and SERMODE are provided on the Pro-Log Utilities Disk.

MODE is an MS-DOS command provided in the "msutils" subdirectory. It can be used to change the communications parameters of COM1. (A 7312-1, 7314A-1, or 7350 card must be installed so that the RS-232-C port on the 7888 card can be used as the COM1 device.) The MODE command can also be used to redirect the printer output to COM1 or change the parallel printer mode (when the 7392 is installed). When the 7350 EGA/Keyboard Card is installed, it can also be used to set display modes.

SERMODE is a Pro-Log utility provided in the "stdutils" subdirectory. SERMODE allows you to redirect the PRN or AUX device I/O to the serial ports (CON or SERx) provided by the 7312-1 or 7314A-1 communications card. It also sets options and initializes the CON and SERx devices provided on the 7312-1 or 7314A-1 communications card.

6.2 BLOCK DEVICES

Block devices are devices such as disk and tape drives that transfer data in multiple-byte blocks. System 2 supports semiconductor disks as well as floppy and hard disk drives. Figures 6-2 and 6-3 summarize the default drive letter assignments used in both models of System 2.

BLOCK DEVICE	PHYSICAL LOCATION	TYPE OF DRIVE	CAPACITY	STANDARD OR OPTION	STORAGE FOR:
Drive A	7713-01/02 Ext. 3.5" Drive Ext. 5.25" Drive	ROM Disk Floppy Disk Drive Floppy Disk Drive	256K Bytes 720K Bytes 360K Bytes	Standard 7391-01/903185 7391-01/903197	MS-DOS 3.2 Programs and Data Programs and Data
Drive B	7713-01 7713-02 Ext. 3.5" Drive Ext. 5.25" Drive	User-Supplied ROM Disk Battery-Backed RAM Disk Floppy Disk Drive Floppy Disk Drive	256K Bytes 128K Bytes 720K Bytes 360K Bytes	Standard Option 020 7391-01/903185 7391-01/903197	Programs and Data Programs and Data Programs and Data Programs and Data
Drive C	3.5" Hard Drive	Winchester Disk Drive	20 Megabytes	Option 021	Programs and Data
Drive D	Pro-Log Reserved	-	-	-	-
Drive E	Pro-Log Reserved	-	-	-	-
Drive F	Pro-Log Reserved	-	-	-	-
Drive G	Pro-Log Reserved	-	-	-	-
Drive H	7713-03/04	ROM Disk	256K Bytes	Option 025/026	MS-DOS 3.2
Drive I	7713-03 7713-04	User-Supplied ROM Disk Battery-Backed RAM Disk	256K Bytes 128K Bytes	Option 025 Option 026	Programs and Data Programs and Data
Drives H-U	7714-01 7715-01 7715-02 7715-03	ROM Disk RAM Disk Battery-Backed RAM Disk Battery-Backed RAM Disk	512K Bytes 256K Bytes 128K Bytes 256K Bytes	7714-01 7715-01 7715-02 7715-03	Programs and Data Programs and Data Programs and Data Programs and Data
Drives V-Z (Available for Installable Block Drivers)	Optional I/O	-	-	-	-

Figure 6-2. DOS Block Devices (Model 10)

BLOCK DEVICE	PHYSICAL LOCATION	TYPE OF DRIVE	CAPACITY	STANDARD OR OPTION	STORAGE FOR:
Drive A	Internal 3.5" Drive	Floppy Disk Drive	720K Bytes	Standard	MS-DOS 3.2
Drive B	Ext. 3.5" drive Ext. 5.25" Drive	Floppy Disk Drive Floppy Disk Drive	720K Bytes 360K Bytes	7391-01/903185 7391-01/903197	Programs and Data Programs and Data
Drive C	3.5" Hard Drive	Winchester Disk Drive	20 Megabytes	Option 024	Programs and Data
Drive D	Pro-Log Reserved	-	-	-	-
Drive E	Pro-Log Reserved	-	-	-	-
Drive F	Pro-Log Reserved	-	-	-	-
Drive G	Pro-Log Reserved	-	-	-	-
Drive H	7713-03/04	ROM Disk	256K Bytes	Option 025/026	MS-DOS 3.2
Drive I	7713-03 7713-04	User-Supplied ROM Disk Battery-Backed RAM Disk	256K Bytes 128K Bytes	Option 025 Option 026	Programs and Data Programs and Data
Drives H-U	7714-01 7715-01 7715-02 7715-03	ROM Disk RAM Disk Battery-Backed RAM Disk Battery-Backed RAM Disk	512K Bytes 256K Bytes 128K Bytes 256K Bytes	7714-01 7715-01 7715-02 7715-03	Programs and Data Programs and Data Programs and Data Programs and Data
Drives V-Z (Available for Installable Block Drivers)	Optional I/O	-	-	-	-

Figure 6-3. DOS Block Devices (Model 20)

6.2.1 Virtual (Semiconductor) Disk

A virtual disk is an I/O-mapped bank-selected memory card programmed to emulate a mechanical disk, such as a floppy or fixed (hard) disk. System 2 supports three types of virtual disks: ROM, RAM, and battery-backed RAM. Virtual disks are accessed the same as mechanical disks. For example, the MS-DOS commands such as DIR (list directory filenames) and CHKDSK (report disk status) work the same on a virtual disk as on a mechanical disk drive.

DIFFERENT TYPES OF VIRTUAL DISKS

ROM Disks

ROM disks are used to contain the operating system and application programs. Depending on the configuration of the system you ordered, it may include a 7713 System Disk Card. The 7713 provides the MS-DOS operating system on ROM disk drive A (7713-01 and 7713-02) or ROM disk drive H (7713-03 and 7713-04).

The 7713 card may also contain four empty 28-pin JEDEC sockets that can accommodate a 256K-byte user-supplied ROM disk drive B (7713-01) or ROM disk drive I (7713-03). In some configurations of System 2, this second ROM disk is replaced by a 128K-byte battery-backed RAM disk drive B (7713-02) or drive I (7713-04).

A ROM disk is similar to a write-protected floppy disk in that files can be read from the disk but not written to it.

Data cannot be written to a ROM device through the operating system. For example, the following error message is displayed when an attempt is made to write to ROM disk A:

```
Write fault error writing drive A
Abort, Retry, Ignore?
```

Respond to this message with <a> to abort.

You can place your application programs in ROM disk using the ROMDISK and SENDFILE utility programs supplied on the STD-DOS Utility Disk. These utility programs allow you to transfer your programs to 27C512 CMOS EPROM (or equivalent) devices through a PROM Programmer. Chapter 8 includes instructions on how to use these two utility programs.

Optional ROM-Disk Expansion

If you need additional ROM disks, the 7714 Semiconductor Disk EPROM Card can be installed.

The optional 7714 Semiconductor Disk EPROM Card allows you to expand the number of ROM disks in your target system. Each 7714 card provides support for up to 512K bytes of user-supplied ROM, using eight 27C512 CMOS EPROM (or comparable ROM) devices.

A rotary drive letter select switch allows each 7714 to be accessed as a unique drive letter. These optional 7714 ROM Disks can be accessed starting at drive letter H. If you ordered Option 025 or 026 (7713-03 or 7713-04) installed, the optional ROM disks can be accessed starting at drive letter J. Up to 14 cards can be added to System 2, for a total of 7 megabytes of ROM disk storage.

RAM Disks

A virtual RAM disk works like a mechanical disk that is not write protected. Files can be transferred to and from a RAM disk. Because RAM is volatile, data stored on a RAM disk is temporary. That is, when power is removed from the system the files are lost.

RAM disks are optional to System 2 but can easily be added to the system by installing the 7715-01 Semiconductor Disk RAM Card.

Each 7715-01 card provides 256K bytes of RAM, using eight 32 by 8-byte CMOS RAM devices. These cards have an onboard capacitor that maintains the disk data for a minimum of 30 minutes (typically for days). If you need to battery-back the data on these disk cards for a longer period of time, a battery can be connected to the VBATT line on the STD BUS.

A rotary drive letter select switch allows each 7715-01 to be accessed as a unique drive letter. These optional 7715-01 RAM Disks can be accessed starting at drive letter H. If you ordered Option 025 or 026 (7713-03 or 7713-04 installed), the optional RAM disks should be accessed starting at drive letter J. Up to 14 7715 cards can be added, for a total of 3.5 megabytes of RAM disk storage.

Battery-Backed RAM Disks

Battery-backed RAM disks are also optional to System 2. They operate the same as RAM disks but provide the capability to store your files even after power has been removed from the system. Files copied to a battery-backed RAM disk are typically retained for five years (cumulative time in the absence of Vcc). Battery-backed RAM disks provide an alternative to transferring files to PROM for permanent storage, especially when the system is operational 24 hours a day.

A 128K-byte battery-backed RAM disk drive is provided on the 7713-02 and 7713-04 System Disk cards. This disk is accessed as drive B on the 7713-02 and is accessed as drive I on the 7713-04. A write-protect switch on the 7713 card provides the same function as the write-protect tab on a floppy disk. It prevents transferring or removing files to or from the disk.

If your system requires additional battery-backed RAM disks, you can purchase and install the 7715-02 or 7715-03 Battery-Backed RAM Disk Expansion cards. The 7715-02 provides a 128K-byte battery-backed RAM disk, while the 7715-03 provides a 256K-byte battery-backed RAM disk.

A write-protect switch on the 7715-02/03 cards protects files from being transferred or removed. Up to 14 7715-02 or 7715-03 cards can be added to the system. The rotary switch on the front edge of the card selects the drive letter (H-U) by which the disk is addressed.

RAM, Battery-Backed RAM, and ROM Expansion Disks in Combination

Up to fourteen 7714 or 7715 Disk Expansion cards can be installed in your System 2. Each of these cards must be selected as a unique drive via the rotary switch provided on the edge of the card. If your system includes a 7713-03 or 7713-04 card, the disk expansion cards should be accessed starting at drive letter J. If a disk expansion card is selected as drive H and either of these 7713 cards is installed, the system will not boot the operating system.

6.2.2 Mechanical (Floppy and Hard) Disk Drives

FLOPPY DISK DRIVES

System 2 can address two 3.5-inch or 5.25-inch IBM PC/XT-compatible floppy disk drives via the 7391-01 Floppy Disk

Controller Card. When a 7713-01 or 7713-02 is not installed in the system, the first floppy drive is always accessed as drive A. If a second floppy drive is installed, it is always accessed as drive B. (The drive select jumpers of the first drive should be set for DS0. The second floppy drive should respond to DS1.)

If a 7713-01 is installed in the system (the standard configuration of Model 10), only one floppy drive should be added to the system. It will be accessed as Drive B. It is not recommended that floppy drives be added to systems that contain the 7713-02 (Model 10 Option 020).

System 2 Model 20 includes an internal 3.5-inch floppy disk drive A and MS-DOS operating disk. The 3.5-inch floppy disks used with Model 20 provide 720K bytes (formatted) of storage. You can add an additional 3.5-inch or 5.25-inch drive B external to the card rack.

For more information on the 7391-01 Floppy Disk Controller Card, refer to the *7391 User's Guide.*

HARD DISK DRIVES

If you ordered Model 10 Option 021 or Model 20 Option 024 at the time you placed your order for System 2, the 7390-02 SCSI Host Adapter Card and hard disk drive are installed in the system card rack. The hard (Winchester) disk drive installed is a 3.5-inch Rodime RO 652A drive (or one with equivalent specifications). It provides 20 megabytes of storage (formatted). If installed, the Winchester drive is always accessed as Drive C. When shipped in this configuration, the system files have been transferred to the hard disk and the system can boot from Drive C.

For more information on the 7390-02 SCSI Host Adapter Card, refer to the *7390 User's Guide* shipped with Option 021.

6.3 INSTALLABLE DEVICE DRIVERS

STD-DOS allows the user to add new character and block device drivers to the system through the use of installable device drivers. These are files written by the user that allow DOS to access user-added hardware. Specifications for these files are given in the *DOS Technical Reference.*

Installable device drivers are added to the system using the DEVICE command:

```
DEVICE=[d:][directory]filename[.ext]
```

The device command must be included in the CONFIG.SYS file on the drive from which the operating system boots. Do not use the ANSI.SYS device driver in the CONFIG.SYS file in System 2 if you are using the STD.EXE utility. The two programs are not compatible and will lock up System 2.

Chapter 7

Preparing Application Programs

7.1 PROCEDURE

1. **Select an editor/word processor to create the source program.**

   Choose from a wide variety of readily available PC-compatible editing and word processing packages. For your convenience, Pro-Log distributes the popular and powerful BRIEF Editor, sourced by UnderWare.

2. **Select the language.**

   The large selection of readily available PC-compatible languages allows you to write your control programs in the language that suits your application. Pro-Log has tested, recommends, and distributes Microsoft's Macro Assembler, C Compiler, and BASIC languages. Our STD-LIB software library (version 2.0 or later) provides I/O, timing, and memory diagnostic routines that can be linked to the Microsoft compilers and assemblers that we distribute.

3. **Write the application program using the selected editor.**

   The program you create with the editor/word processor is sometimes referred to as the "source program" because it is the source from which the executable file is generated.

4. **Assemble or compile the application program.**

   Unless you are using a BASIC interpreter, the program is assembled or compiled into a relocatable object file. In most cases, a program is normally written in several object files and/or combined with modules from an application library at link time.

5. **Link the program modules.**

   The linker creates executable programs from object modules generated by the assembler or by high-level language compilers. A linker combines code and data in the object files and searches the named libraries to resolve external references to routines and variables. A library, such as STD-LIB, is simply a collection of compiled or assembled functions that provide a common set of useful routines.

6. **If you have used the PC to develop your program, you will need to load the program in System 2 for testing and debugging.**

   If System 2 and the PC contain compatible floppy disk drives (IBM PC/XT-compatible 5.25- or 3.5-inch), your program can be loaded from floppy disk. Otherwise, your software can be downloaded to the hard disk, RAM disk, or battery-backed RAM disks in System 2.

   When the PC is connected to the system through the console, you can use the STD.EXE utility program or Microstuf's CROSSTALK package to download files from the PC to a disk drive in System 2.

   If you are using the PC for program development and have a 7350-01 EGA/Keyboard Card in System 2, the COM1 port of the PC should be connected through a null modem to the COM1 port of the 7888 CPU Card in System 2. The XXMT.EXE and XRCV.EXE utility programs can be used to transfer files to and from the PC and System 2 through these COM1 ports. The STD.EXE, XXMT.EXE, and XRCV.EXE programs are on the STD-DOS Utilities Disk provided with each system. Refer to Chapter 9 for details on how the XXMT.EXE and XRCV.EXE communication utilities can be used.

7. **Test and debug your application program.**

   Perform system test and debug using the resident debugger (DEBUG) or any PC-compatible high-level language debugger. DoubleDOS, from SoftLogic Solutions distributed by Pro-Log, can simplify program debugging by allowing System 2 or the PC to run two programs concurrently. By pressing <Alt> <Esc>, you can shift from the display generated by the debugger to the display generated by an editor, compiler, or assembler.

   If you're using the PC for program development, DoubleDOS allows you to run the STD.EXE or CROSSTALK programs simultaneously with an editor, compiler, or assembler. The STD.EXE and CROSSTALK programs allow you to view the display generated by the resident debugger in System 2 while the BRIEF editor (running in System 2 or the PC) displays the source code of the program being debugged.

8. **Once the application program is fully tested and proven, the file can be transferred to a ROM or RAM disk or to a floppy or hard disk.**

   If your system uses floppy and/or hard disks, you can store your application programs on these disks. To store your programs on a ROM disk, use the ROMDISK utility program to format your executable file. The SENDFILE utility program can then be used to download the file to a PROM Programmer. After the files have been transferred to EPROM,

the devices can be installed in the 28-pin JEDEC sockets in the 7713-01 or 7713-03 System Disk Cards or the optional 7714 ROM Disk Card.

See Chapter 8 for details on how to run the ROMDISK and SENDFILE utility programs. These programs are provided on the utility disk supplied with the system. The ROMDISK and SENDFILE utility programs can be run on an IBM PC, PC/XT, PC/AT, a hardware-compatible PC, or a comparably equipped System 2. If you're using a PC, it should have the Asynchronous Adapter Card (or 8250 hardware equivalent) configured as COM1, to provide an RS-232-C interface to the PROM Programmer. If you are using System 2, the COM1 port on the 7888 CPU card should be connected to the PROM Programmer for downloading the file(s) to create your ROM disks.

With MS-DOS resident in System 2, running the application program(s) is as easy as typing the filename, just as you would execute an application program on your PC. If the system is to operate unattended and the program is to run from power-up, you need to create a "startup" file.

9. Create a batch file to cause the system to run programs from power-up.

The target system can also operate from power-up without requiring any user interface. A terminal or PC need only be reconnected when system functions require modification. Executing the application program as part of the power-up sequence is simply a matter of including the application filename in a "startup" file. The type of startup file you create depends on the type of drive from which the system boots.

If you're booting the system from ROM disk, the AUTOEXEC.BAT file on drive A (or drive H) searches drive B (or drive I) for a file called "STARTUP" during power-on. If any filename called STARTUP is found (it can be either STARTUP.BAT, STARTUP.EXE, or STARTUP.COM), it is immediately executed.

If you have purchased the hard disk option, the system can boot from the hard disk drive (drive C) unless you have removed the system files. Every time you boot the system, MS-DOS searches for a file named AUTOEXEC.BAT in the root directory of the drive from which the system has booted. If you are booting the system from hard disk drive C, include the application filename in the AUTOEXEC.BAT file. It will be automatically executed when you start or restart the system from drive C. For more information on batch files, refer to the *DOS Reference* manual.

Chapter 8

Transferring Executable Files To A ROM Disk

8.1 THE ROMDISK UTILITY PROGRAM

The ROMDISK.EXE Utility Program transforms your application files into binary (.BIN) or hex (.HEX) files that not only contain the application file data but also the file allocation table (FAT) and directory information for the application files. Formatting the application files this way allows the files, when they are downloaded to a PROM Programmer and placed in Pro-Log ROM disk card, to appear to MS-DOS as a write-protected floppy disk.

The files generated from the ROMDISK program can be downloaded from a directory in your System 2 or PC to your PROM Programmer using the SENDFILE Utility Program. Information on the SENDFILE program is provided in the last part of this chapter.

The ROMDISK and SENDFILE utility programs run on System 2 or a PC. They are provided on the STD-DOS Utilities disk and the 3.5-inch System disk shipped with Model 20.

We recommend you read this chapter before actually executing the ROMDISK program.

The following step-by-step procedure illustrates how the ROMDISK Utility Program could be run. The underlined characters are the characters you type from the console keyboard. The response of the ROMDISK program is in **bold** type.

Step 1: Power-on System 2 or the PC used to run the ROMDISK program. Insert the 3.5-inch system disk or the STD-DOS Utilities Disk into the floppy drive of System 2 or your PC. For this example, we are assuming the STD-DOS Utilities Disk has been inserted into drive B of the PC.

Using the "CHDIR" or "CD" command, make the root directory of the STD-DOS Utilities Disk in drive B the current directory.

CD B: <CR>

If the ROMDISK.EXE file resides on another directory, select the appropriate directory.

Step 2: Start the ROMDISK Utility by typing:

B>ROMDISK<CR>

ROMDISK version 2.00
(C)Copyright 1985, 1986 Pro-Log Corp.

If a <CR> is entered in response to prompts,
the default data within [ ] will be accepted.
A <^C> entered, will terminate ROMDISK.
Data within < > is optional.

Step 3: The ROMDISK program asks if the ROM disk you are generating is to be used to boot the system. In most cases, the ROM disk is used for storage of application programs and is not used to boot the system. If this is the case, follow the procedure outlined in Step 3a and skip 3b. If you are using the ROMDISK Utility to customize ROM disk A (or ROM disk H), proceed with Step 3b.

Step 3a: If the ROM disk will be used for program storage, press the ENTER key after the prompt, as illustrated below.

ROMDISK to be used to boot from? <Y>=yes [N]=no <CR>

Step 3b: If you are using the ROMDISK program to customize the ROM disk containing the operating system, you should save the three factory-programmed EPROMs that comprise the as-shipped version of ROM disk A (or ROM disk H).

At a minimum, the system files IO.SYS, MSDOS.SYS, COMMAND.COM, and @@XFER@@.EXE must be placed in ROM disk A (or H) of System 2. These files reside in the MSDOSSYS subdirectory of the STD-DOS Utilities Disk root directory. The system files are also included in the MSDOSSYS

subdirectory on the 3.5-inch operating disk provided with Model 20.

If you are creating a new ROM disk A, respond to the prompts in the following way:

ROMDISK to be used to boot from? <Y>=yes [N]=no Y

Required SYSTEM files:

IO.SYS
MSDOS.SYS
COMMAND.COM
@@XFER@@.EXE

Enter SYS files <drive:> <path> [C] B:\MSDOSSYS <CR>

The SYSTEM files are located within the MSDOSSYS subdirectory. If you have copied the ROMDISK file from the STD-DOS Utilities Disk to another drive or directory, type the drive letter and path where the SYSTEM files are located.

Step 4: Next, the program prompts you for the names of the files you want stored in the ROM disk.

Enter SOURCE file(s) <drive:> <path> filename<.ext> (CR=end)

Type in the name(s) of the executable file(s) to be placed in your ROM disk. The maximum number of files is 128. In the example below, the files asmdemo.exe in subdirectory c:\asm88, basdemo.exe in subdirectory c:\msbasc, and cdemo.exe in subdirectory c:\msccomp will be placed in this ROM disk.

c:\asm88\asmdemo.exe <CR>
c:\msbasc\basdemo.exe<CR>
c:\msccomp\cdemo.exe <CR>
<CR>

NOTE: The ROMDISK utility only creates a list of the files, i.e., it does not transfer any files at this time.

Step 5: Next, the program prompts you for a four-character "identifier." These characters are the last four characters in the name of the hex or binary file to be downloaded to the PROM Programmer. (You will be prompted to select a binary or Intel hex format in the next step.) In this example, we have used "test" as the four-letter identifier.

Enter ROMDISK file(s) name [ROM0-7] <name> test

In this example, the name of the file to be downloaded to the first ROM is "ROM0TEST.HEX" or "ROM0TEST.BIN," depending on your selection in the next step. If the size of the program(s) entered in Step 2 exceeds the capacity of one EPROM, the program makes up to eight files. Each of these files is downloaded to one device. If a second EPROM were required, the name of the file to be downloaded to the second EPROM would be "ROM1TEST.HEX" or "ROM1TEST.BIN," and so on. This would be the case of the four system files, required to boot the MS-DOS operating system, were to be resident on a ROM disk along with the application files.

Step 6: The program asks if the file to be downloaded to the PROM Programmer should be in hexadecimal or binary format. If the PROM Programmer accepts a hex file, press the ENTER key. If the file is generated as a HEX file, it will have a .HEX extension. If it is a binary file, it will have a .BIN extension.

Enter ROMDISK file(s) type <B>=bin or [H]=hex <CR>

Step 7: The ROMDISK program prompts you for the destination (drive and path) for your ROM file(s). You can then download the file, via the SENDFILE Utility Program, to your PROM Programmer from this directory. (The ROMDISK program is not prompting you for the drive letter of the ROM disk you are creating.) If the file is to be placed in your current directory, just <ENTER>.

Enter ROMDISK files destination <drive> \ <path> [C] <CR>

The system then responds with:

```
BUSY...Making files for ROMDISK

Saving file for ROM 0
ROMDISK function successful

Source Files copied:
ASMDEMO.EXE
BASDEMO.EXE
CDEMO.EXE

ROMDISK files saved:
ROM0TEST.HEX
```

NOTE: If the ROM disk will be used to boot the system, the program would have responded with:

```
BUSY...Making files for ROMDISK

Saving file for ROM 0
Saving file for ROM 1
ROMDISK function successful

Source Files copied:
IO.SYS
MSDOS.SYS
COMMAND.COM
@@XFER@@.EXE
ASMDEMO.EXE
BASDEMO.EXE
CDEMO.EXE

ROMDISK files saved:
ROM0TEST.HEX
ROM1TEST.HEX
```

Step 8: Use the SENDFILE Utility Program to download the file(s) to a PROM Programmer.

8.2 DOWNLOADING YOUR BINARY OR HEX FILES TO A PROGRAM PROGRAMMER

8.2.1 The SENDFILE Utility Program

The SENDFILE Utility Program downloads files generated from the ROMDISK Utility Program to a PROM Programmer. If the PROM Programmer you have purchased includes a program that provides the capability to download files to it from the PC, it should be used instead.

Follow the instructions listed below to use the SENDFILE Utility for downloading your files to a PROM Programmer. The underlined characters are the characters you might type when running the program. The response of the SENDFILE program is in **bold** type.

Step 1: Start the SENDFILE Utility

B>SENDFILE <CR>

SENDFILE version 2.0
(C)Copyright 1986 Pro-Log Corp.

Press <CR> to accept data within brackets [ ]
Press <^C> to exit SENDFILE
Data within < > is optional

Step 2: The program requests the name of the file to be downloaded to the PROM Programmer.

Enter file <drive> <path> filename.ext
rom0test.hex

The ROM0TEST.HEX file that was generated in the ROMDISK Utility Program example is downloaded to the programmer.

Step 3: Select the DOS device to be used to download the file. The default device is the COM1 port.

****** RS232 OUTPUT PORT INITIALIZATION PARAMETERS ******

Output Port selections
Key 0 = COM1
Key 1 = SER1
Key 2 = SER2
Key 3 = SER3
Key 4 = SER4
Select Output Port [COM1] <CR>

Step 4: Select the baud rate for the device selected above. In this example, we are initializing the COM1 port for 9600.

Baud Rate selections
Key 0 = 9600
Key 1 = 4800
Key 2 = 2400
Key 3 = 1200
Key 4 = 600
Key 5 = 300
Key 6 = 150
Key 7 = 110
Select Baud Rate [9600] <CR>

Step 5: Select the parity. In this example, a parity bit is not used.

Parity Status selections
Key 0 = None
Key 1 = Even
Key 2 = Odd
Select Parity Status [None] <CR>

Step 6: Select the number of stop bits required. The default value is to send with one stop bit.

Stop Bits selections
Key 0 = 1 stop bit
Key 1 = 2 stop bits
Select Stop Bits [1] <CR>

Step 7: Select the number of bits in a character. A transmitted character consists of 8 bits.

Character Length selections
Key 0 = 8 bits
Key 1 = 7 bits
Select Character Length [8] <CR>

Step 8: The SENDFILE program can download hexadecimal or binary files. In this example, we are downloading the hex file ROM0TEST.HEX. If the file to be downloaded is in a binary format, simply press <1>.

****** SEND FILE PARAMETERS ******

File Type selections
Key 0 = HEX
Key 1 = BIN
Select File Type [HEX] <CR>

Step 9: Select the size of a block to be downloaded. This block size is usually dependent upon the size of the buffer in the programmer. A 64K-byte block is downloaded in this example.

Send Block Size selections
Key 0 = 64K
Key 1 = 32K
Key 2 = 16K
Key 3 = 8K
Key 4 = 4K
Key 5 = 2K
Key 6 = 1K
Select Send Block Size [64K] <CR>

Step 10: The program then displays the following message and waits for you to set up the PROM Programmer:

Prepare destination unit to receive...When ready, hit any key

In this example, the PROM Programmer should be set up to receive a hexadecimal file at 9600 baud, with 8 data bits, no parity, and 1 stop bit. It should have a data buffer large enough to store a 64K-byte file.

After you have connected the PROM Programmer to the COM1 port of the PC and have set the communication parameters as listed above, press any key. The system should respond with:

BUSY...Sending block...
Number of Bytes Sent (hex) = 00XX

When the file has been downloaded to the programmer, the system responds with:

File send complete

Step 11: Remove the EPROMs from the programmer and install them in the 7713 or 7714 ROM Disk cards. Refer to the *7713/7714/7715 User's Guide* for instructions on installing the EPROMs in the ROM Disk Cards.

Chapter 9

Transferring Files Between System 2 and Other Systems

9.1 ADDITIONAL SYSTEM 2 COMMUNICATIONS UTILITIES

In addition to STD.EXE, ROMDISK.EXE, and SENDFILE.EXE, several other Pro-Log utility programs and files are supplied with System 2. These programs are used for communications and file transfer between System 2 and other systems, such as another System 2 or an IBM PC. They were written specifically to be used with CROSSTALK, an industry-standard data communications software package, but some are usable with other communications software and as standalone utilities. CROSSTALK can be used in System 2 only when the RS-232-C port on the 7888 CPU card is used as the console or COM1 port.

9.2 COMMUNICATIONS USING THE 7888 CONSOLE PORT

9.2.1 CROSSTALK Script and Command Files

The XTALK directory on the Pro-Log Utilities disk contains script and command files written by Pro-Log that are used with CROSSTALK data communications software. They are used by CROSSTALK to configure the PC's RS-232 communications port when the PC is used as the console for System 2. Refer to the CROSSTALK user's manual for more information on using command and script files.

The command files (*.XTK) set CROSSTALK parameters, make the "connection" to System 2, and initiate file transfer. PROLOG.XTK is used when the PC's COM port is connected directly to the 7888 serial port (through a null modem connector, since both serial ports are DTE). PLMODEM.XTK is used when the 7888 console port is accessed through a modem.

9.2.2 System 2 Directly Connected to Another System

PROLOG.XTK is similar to STD.EXE except that wildcard characters cannot be used in the filename and hence each file must be downloaded one at a time. PROLOG.XTK sets up the following CROSSTALK parameters for local connection to the STD system:

INFilter	Off	OUtfilter	Off
FLow	SQ	Handshake	On
SPeed	19200		

Alt-F1	help
Alt-F3	copy file from PC to System 2
Alt-F4	copy file from System 2 to PC

Function keys Alt-F3 and Alt-F4 initiate the file transfer functions, just as with the STD.EXE utility. To use them, follow the prompts given on the CROSSTALK command line. File transfer is easiest when the file to be transferred is in the PC's default directory, along with the .XTK and .XTS files. (If the file is in another directory, the default directory must be changed, and then changed back to the original default before another file transfer.)

On the System 2 end, the executable files XXMT.EXE and XRCV.EXE must be either on the System 2 default directory or in the path. Also, files can only be transferred when the 7888 onboard serial port is the console interface. The file transfer functions will not work if a 7312-1 or 7314A-1 serial port is used as the console.

The script files (*.XTS) are used internally by CROSSTALK for file transfer after communications have been established. They are never called directly by the user and should be in the default directory along with all other CROSSTALK files. The script files are:

prolog.xts - sets the last command in prolog.xtk to go local line

xrcv.xts - issues XRCV command to System 2 and XMODEM to CROSSTALK for file transfer

xxmt.xts - issue XXMT command to System 2 and RXMODEM to CROSSTALK for file transfer

Again, refer to the CROSSTALK user's manual for more detailed information on the use of command and script files.

9.2.3 System 2 Remotely Connected to Another System Through a Modem

Before PLMODEM.XTK can be used, an installable console device driver, TELE.SYS, must be installed. TELE.SYS allows the 7888 serial port to be used with a modem. The console can then be a remote terminal or PC connected to System 2 over a phone line, allowing remote operation of the STD-DOS system.

This driver is loaded at power-on by specifying it in the "device" statement of the CONFIG.SYS file. The "password" and the "password fail retry count" are specified on the same line. For example, if the following line is entered in CONFIG.SYS:

```
device = tele.sys saysame 3
```

"saysame" will be the password and three retries are given before disconnect if the password is incorrect. The password is case insensitive. If a password or retry number is not specified, the defaults are no password and three retries.

The 7888 serial port should be connected to a Hayes 1200B or compatible modem. **NOTE**: The RC716 cable can be used for this connection; the RC713 Null Modem MUST NOT BE USED. The following handshake lines are used:

Incoming:
DCD (data carrier detect)
= 1 indicates valid line for modem or valid local connection
CTS (clear to send)
DSR (data set ready)

Outgoing:
DTR (data terminal ready) = 1 always
RTS (request to send) = 1 always

The 1200B modem's internal switch settings should be as follows:

SWITCH	POSITION	FUNCTION
1	up	monitor the actual state of DTR
2	up	word result codes displayed
3	up	result codes not sent
4	down	characters not echoed in command state
5	up	auto-answer
6	up	RS-232 Carrier Detect is not always on
7	up	single line RJ-11 phone jack
8	down	enables commands
9	up	compatible with Bell 103/212A modems
10	up	modem reset when turned on

After TELE.SYS is installed, CROSSTALK can be used on the PC to dial and make the connection to System 2 using the PLMODEM.XTK command file as described above. (You should modify the PLMODEM file to include the correct phone number.) Other suitable communications software may be used instead of CROSSTALK, such as PFS Access or PC\InterComm. The baud rate used is 1200 baud.

After the connection is made, you are prompted to enter the password. If the correct password is not entered, System 2 terminates the call after the specified number of retries. If the correct password is entered, the sign-on message "Welcome to the Pro-Log system!" is displayed. You should then press <ENTER>; this flushes the TELE.SYS driver's buffers of any characters left over from the last call, or any spurious characters caused by disconnect or connect operations. System 2 should respond (assuming it is at the command line and not executing any program) with "Bad command or file name" followed by the command line prompt.

The PC is then the console for System 2.

9.3 COMMUNICATIONS USING THE SYSTEM 2 COM1 PORT

In addition to their use with CROSSTALK, the XXMT.EXE and XRCV.EXE utilities can be executed from the DOS command line to transmit a file between any two MS-DOS systems that have PC-compatible COM ports. For example, between an STD-DOS System 2 and an IBM PC. A 7350-01 card or either a 7312-1 or 7314A-1 card must be installed in the System 2 so the port on the 7888 card will be assigned COM1.

The file is transferred across an RS-232 serial link between the COMx serial port on each system. Since PC-compatible COM1 or COM2 ports are always configured as DTE, the RC713 Null Modem or equivalent must be used when connecting the two COM ports. Inside the RC713, RS-232 lines 2 and 3, 4 and 5, and 20 and 6-8 (lines 6 and 8 shorted together) are swapped. To transfer a file from System A to System B, the XXMT.EXE utility is executed on system A, and the XRCV.EXE is simultaneously executed on System B.

There are two ways to execute the utilities, either with or without parameters on the command line:

XRCV FILENAME

or

XRCV FILENAME PORT SPEED MSGFLAG

The same syntax is used with XXMT.EXE. If parameters are not given on the command line, COM1 is used, the current baud rate of COM1 is the speed, and no message displayed. If parameters are specified, PORT defines the COM port that will be used for the file transfer: if PORT = 0, COM1 will be used; if PORT = 1, COM2 will be used. SPEED specifies the baud rate: standard baud rates from 300 to 19200 baud can be used. If MSGFLAG = 1, a description of the file transfer progress is displayed on the console; if MSGFLAG = 0, nothing is displayed during the transfer.

If CROSSTALK is used at the PC end, operation is exactly as described above, including file transfer operations. If a different communications package is used, refer to its user's manual for instructions. File transfer may be possible with other packages that support the XMODEM protocol, using the XXMT.EXE and XRCV.EXE utilities on System 2. The appropriate command line, either "XXMT FILENAME" or "XRCV FILENAME", must be given to System 2, and then the user should initiate his/her communications software's XMODEM receive or transmit function

Product	EGA/Keyboard Required	System Memory Requirements (KB) 128	384	640	Requires/ Supports 8087
Microsoft Macro Assembler	No	X			Supports
Microsoft C Compiler 1	Sometimes 2		X		Supports
Microsoft BASIC Compiler	No	X			
Microsoft BASIC Interpreter	No	X			
Solution Systems BRIEF	Yes		X		
National Instruments GPIB	No	X			
CROSSTALK Communications DCA	Yes	X			
Softlogic Systems DoubleDOS	Yes	X			
Microsoft QuickBASIC	Yes		X		Ver 3+ Supports
Equinox Loopworks	Yes			X	Required
Baker & Rabinowitz PAX	No	X			

[1]Pro-Log currently distributes version 4.0 of the Microsoft C Compiler. Pro-Log previously distributed version 3.0 of the Microsoft C Compiler.

[2]The codeview debugger, included with the Microsoft C Compiler (version 4.0 and above), requires the installation of the 7350 EGA/Keyboard card to be fully utilized. Otherwise CODEVIEW must be used in sequential mode.

Pro-Log reserves the right to make changes to any products herein to improve reliability, function, or design. Pro-Log does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others.

PRO-LOG CORPORATION
2560 Garden Road, Monterey, California 93940
Phone: (408) 372-4593, (800) 538-9570, Telex: 171879
TWX: 910-360-7082, FAX: (408) 646-3517

113657-001 8\87

STD-DOS SYSTEM 2 USER'S GUIDE ADDENDUM

READ THIS FIRST

Some changes have been made to System 2 since the printing of the *System 2 User's Guide*. Please read the following carefully then file this Addendum in the *System 2 User's Guide* binder.

1.0 Option 025, MS-DOS ROM disk, is now standard on Model 10

System 2 Model 10 option 025, which changes the MS-DOS ROM disk from drive A to drive H, is now shipped with all Model 10s. Moving the MS-DOS ROM disk to drive H leaves drive A for use with a floppy disk drive since only drives A and B can be used for floppy drives.

2.0 Transferring data on a 3.5-inch floppy disk to an IBM system

IBM systems expect to find the letters "IBM" in the first sector on a 3.5-inch disk. The System 2 format program does not add this information to the disk; therefore, the IBM system does not view the System 2 formatted disk as a valid readable disk. Floppy disks formatted on an IBM system can be read by both IBM and Pro-Log systems. To read a 3.5-inch floppy disk formatted on System 2 in an IBM system, use the following DEBUG.EXE commands:

```
Debug <cr>
-L 0 X 0 1 <cr> (where "X" is 0 for drive A, 1 for drive B)
-E 3 <cr>
   49 <space> 42 <space> 4D <space> 20 <space> 20 <space>
   33 <space> 2E <space> 33 <cr>
-D 0
   (you should see the "IBM  3.3")
-W 0 X 0 1 <cr> (where "X" is 0 for drive A, 1 for drive B)
-Q <cr>
```

3.0 Creating a virtual RAM disk from system memory

RAMDRIVE.SYS is a device driver that lets you use part of the system memory as a disk drive. RAMDRIVE.SYS is provided in the MSUTILS subdirectory of the 3.5-inch MS-DOS 3.2 disk (included with Model 20) or the 5.25-inch MS-DOS 3.2 disk (included with Model 10). You may want to use this virtual RAM disk during program development, before programming ROMs for a ROM disk. It can also be used in systems with a single floppy disk and no other read/write disks.

System 2 currently does not allow you to "copy" files from one floppy disk to another when only one floppy drive is installed in the system. The alternative is to copy the files to another read/write drive (floppy, hard, or RAM/battery-backed RAM disks) and then transfer the files back to the floppy drive after the floppy disks have been exchanged. (To copy the entire contents of one floppy disk to another floppy disk, use "diskcopy" instead of copy -- i.e., diskcopy A: A:.)

If there are no other read/write disks in the system, you can create a RAM disk from system memory. The size of the RAM disk depends on the amount of system memory installed in the system.

You need the RAMDRIVE.SYS file on one of the disks in System 2. To allow for MS-DOS, system memory should be at least 83K bytes larger than the RAM disk you are going to create.

To create a 32K-byte RAM disk from system memory, you would:

1. Edit the CONFIG.SYS file (in the root directory of the drive on which you boot the system) so that it contains the command line that installs the device driver RAMDRIVE.SYS.

   For example, change the CONFIG.SYS file on a 3.5-inch MS-DOS floppy disk (shipped with System 2, Model 20) by typing:

   ```
   copy con config.sys <cr>
   device=c:\msutils\ramdrive.sys 32 <cr>
   buffers=20 <cr>
   files=20 <cr>
   Z <cr>
   ```

2. After you have edited the CONFIG.SYS file, reboot the system so the new CONFIG.SYS installs the RAMDRIVE.SYS driver at initialization. The 32K-byte RAM disk will be the next drive letter not reserved as a device. Drive letters A-H are always reserved in System 2. When the system is rebooted, it should give you a message stating which version of Ramdrive executed, the drive letter, and other disk status.

   If ROM or RAM disk cards (7714 or 7715) are installed in the system, the RAM disk created by RAMDRIVE.SYS is accessible as the next drive letter available after the last drive. For example, if the system has H and I drives plus a 7715 card that has been selected as drive R, the next available drive letter would be drive S.

4.0 System requirements for third-party software products

System 2 can execute software written to run under the PC-DOS and MS-DOS operating systems. However, PC-DOS software relies on using specific hardware devices and BIOS interrupts standard in the PC, but optional to System 2. For this reason, the 7350 EGA/Keyboard Card and IBM PC/XT compatible keyboard must be installed in System 2 when using software that is only PC-DOS compatible.

You may need additional system memory if you are editing and compiling programs on System 2. The 7716-01 and 7716-02 RAM Expansion Cards can be installed to provide a total of 384K or 640Kbytes of system memory. These cards can be factory installed or can be purchased and installed at a later time.

Some application software, such as Loopworks, require that the 8087 math coprocessor be installed. Other software packages, such as the Microsoft MacroAssembler, C Compiler, and QuickBASIC (Version 3.0) provide support for the 8087 to speed up mathematical calculations.

The system requirements for each of the software products we distribute are listed in the table on the next page.

Chapter 10
Hardware Reference

10.1 INTRODUCTION

This is the hardware design specification for the System 2 STD BUS product, Model 10 and Model 20. The core system for each model is an STD-DOS computer that includes the MS-DOS 3.2 operating system, a V20 (80C88-compatible) CPU card, a bus termination card, a card rack, and either a semiconductor or floppy disk drive. The disk drive in Model 10 is a 7713 semiconductor ROM Disk card; in Model 20 it is a 3.5-inch floppy disk drive (with controller card).

10.2 IBM PC/XT COMPATIBILITY

System 2 features IBM-compatibility for selected devices at the BIOS and hardware levels, in addition to compatibility at the operating system level. Compatibility at the BIOS level means that BIOS software interrupts are implemented with the same input and output parameters as in the IBM PC BIOS. Compatibility at the hardware level means that physical devices and their I/O port addresses are exactly the same as in the PC (or close enough that all software that directly addresses the PC devices will run with the compatible devices).

Many applications bypass the operating system for certain functions and go directly to the BIOS or to the hardware for increased performance. This is particularly true for clock and timing functions, asynchronous communications, and the console. In System 2, all PC BIOS interrupts are supported, except for the cassette and ROM BASIC Interpreter services. Hardware-compatible devices include the 8254 Counter/Timer, 8259 Interrupt Controller, 8250 UART, and (optional) EGA/Keyboard console and printer interfaces.

Factory-installed options can be ordered with both models of System 2, providing a custom-configured and factory-tested system. These options include memory expansion, hard disk, printer and alternate console interfaces, power supply, and packaging.

10.3 FUNCTIONAL DESCRIPTION

10.3.1 System Specifications

System-level hardware specifications are provided in Chapter 12.

10.3.2 System Components

The following hardware is included with both the Model 10 and Model 20 core systems:

7888 CPU CARD

The 7888-01 includes the V20 CPU (a CMOS 8088-compatible CPU from NEC that features an extended instruction set and faster execution of certain instructions), the BIOS ROM, 128K of static system RAM, and IBM PC hardware-compatible peripherals: 8254 Counter/Timer, 8259 Interrupt Controller, and 8250 UART (COM1/console interface). The socket under the BIOS ROM contains SmartWatch, a Dallas Semiconductor real-time clock with integral lithium battery.

These peripherals provide the necessary hardware compatibility for clock, timing, interrupt, and asynchronous communications functions. CMOS and surface-mount devices reduce power requirements and allow a high functional density on the card. The 7888 meets the STDMG-8088 BUS Specification, Version 2.2.

If you ordered System 2 Option 060, the 7888-01 is replaced with the 7888-03. The 7888-03 provides all the features of the 7888-01 but also includes an 8087 daughter board. The daughter board contains a V20 processor and an 8087 math coprocessor and plugs into the CPU socket.

Refer to the *7888 System 2 CPU Card User's Guide* for additional technical information.

7909A TERMINATION CARD

The 7909A is a bus termination card that provides the passive AC termination described in the *STDMG STD Bus Specification and Practice*, Pro-Log document number 106989, revision E and above. It also provides a +5V indicator LED and a pushbutton for system reset.

RC716 Serial Cable

This transition-to-D connector cable is used to connect the System 2 console port (directly or through a modem) to a PC serial port or terminal. Both male and female D connectors are attached.

RC713 Null Modem

This is an adapter that must be used when the serial port on the 7888 is connected to DTE equipment such as a terminal or a PC serial port.

SYSTEM 2 MODEL 10

In addition to the 7888 CPU Card and 7909A Termination Card, the following hardware is supplied with the Model 10 only:

7713 Semiconductor Disk Card

The 7713 is a paged memory card that provides two solid-state disks. The use of paged memory essentially removes the semiconductor disks from the V20's one-megabyte memory map, allowing full system memory expansion and the use of PC-compatible display adapters and BIOS extensions. The 7713 includes a 192K ROM disk (three 27512-type devices) that contains system files and utilities.

The standard configuration of Model 10 includes the 7713-01 Semiconductor Disk Card. In addition to ROM Disk A, the 7713-01 also provides four empty 28-pin JEDEC sockets for a second (user-supplied) applications ROM disk (up to 256Kbytes, four 27512-devices).

Model 10 with Option 025 is identical to the standard configuration except that the 7713-01 is replaced by a 7713-03. When the 7713-03 is installed, the operating system is contained in ROM Disk H and the user-supplied ROM disk is accessed as drive I.

If you ordered Model 10 Option 020, the 7713-01 is replaced by a 7713-02. The 7713-02 provides ROM Disk A and a 128K battery-backed RAM Disk B. Four Dallas Semiconductor DS1235 32K by 8 nonvolatile static RAMs (or equivalent) provide this battery-backed RAM disk.

If you ordered Model 10 with Option 026, the 7713-01 is replaced by the 7713-04. The 7713-04 provides the operating system in ROM Disk H and the battery-backed RAM Disk is accessed as drive I.

For more detailed specifications on the 7713, refer to the *7713/7714/7715 Solid-State Disk Card User's Guide* included with all configurations of Model 10.

BX06T-01 Card Rack

This is the standard STD BUS card rack shipped with Model 10. The BX06T-01 provides six card slots with connectors on 5/8-inch centers. Three of these slots are occupied by the three STD BUS cards included in the standard system. The other three slots are available for accessory cards.

Model 10 options 001 through 008 provide additional card slots for accessory cards. Depending on the option ordered, 6 to 23 expansion slots are available for factory-installed accessory cards or cards you install in the system.

The card racks can be mounted on horizontal or vertical surfaces or may have been configured for panel or RETMA rack mounting. Option 008 configures the system in a 19-inch RETMA rack and provides 23 user slots (18 slots when the hard disk option is also ordered). If you ordered option 010, your system can be easily mounted to a vertical surface such as the back panel of a NEMA enclosure.

The BX-Series card racks include an integral four-layer motherboard with interleaved signal and ground traces to provide high noise immunity. The cards are supported on three sides by the card rack and are held securely in place by a retaining bar. Terminal blocks are provided on the motherboard for attaching power supplies and powering disk drives.

System 2 Model 10 Options

Refer to Chapter 1 for information on other factory-installed options available for Model 10. Technical information for these options is contained in the user's guide shipped with the option. Information regarding the 7713 Semiconductor Disk Card, 7714 ROM Disk Card, and 7715 RAM Disk Card is included in the *7713/7714/7715 Solid-State Disk Card User's Guide* shipped with all configurations of Model 10.

SYSTEM 2 MODEL 20

In addition to the 7888 CPU Card and 7909A Termination Card, the following hardware is supplied with the Model 20 only:

7391-01 Floppy Disk Controller Card and 3.5-inch Floppy Disk Drive

Model 20 includes one 3.5-inch double-sided, double-density floppy disk drive with mounting bracket and power and signal cables, and a 7391-01 Floppy Disk Controller Card. The 3.5-inch floppy drive is mounted within the card rack on the right-hand side, and power is taken from the STD backplane. This drive is accessed as Drive A.

The signal cable connects the 7391-01 floppy disk controller card to a 3.5-inch floppy disk drive as well as terminates with an expansion port for connecting an external 3.5- or 5.25-inch floppy disk drive via an RC707-1 or RC707-2 interconnect cable.

If you want your system to have two floppy drives, you can add an external 3.5-inch or 5.25-inch floppy drive to the standard configuration of Model 20. The drives are compatible with the IBM 3.5-inch and 5.25-inch standards and provide a formatted capacity of 720K bytes (3.5-inch floppy disk) or 360K bytes (5.25-inch floppy disk).

Model 20 can also contain the operating system in ROM Disk. When Option 025 is ordered, a 7713-03 is installed in the system. The 7713-03 provides the operating system in ROM Disk H and a second user-supplied ROM disk is accessed as drive I. When Option 026 is ordered, a 7713-04 is installed in the system. The 7713-04 provides the operating system in ROM Disk H and a 128K-byte battery-backed RAM disk is accessed as drive I.

BX12T-02

This is the standard STD BUS card rack shipped with Model 20. It has a 9-slot motherboard installed in a 12-slot card rack, leaving room for disk drive mounting. The BX12T-02 provides nine card slots with connectors on 5/8-inch centers. Three of these slots are occupied by the three STD BUS cards included in the standard system. The other six slots are available for accessory cards.

Model 20 options 001 through 005 provide additional card slots for accessory cards. Depending on the option ordered, 9 to 18 expansion slots are available for factory-installed accessory cards or cards you install in the system.

The card racks can be mounted on horizontal or vertical surfaces or can be configured for panel or RETMA rack mounting. Option 005 configures the system on a 19-inch RETMA rack and provides 18 user slots (15 slots when hard disk option is also ordered). If you ordered Option 010, your system can be mounted easily to a vertical surface such as the back panel of a NEMA enclosure.

The BX-Series card racks include an integral four-layer motherboard with interleaved signal and ground traces to provide high noise immunity. The cards are supported on three sides by the card rack and are held securely in place by a retaining bar. Terminal blocks are provided on the motherboard for attaching power supplies and powering disk drives.

10.4 Card Rack Dimensions and Power Connections

System 2 is shipped in a standard Pro-Log BX-Series card rack. Figures 10-1 and 10-2 show which card racks and motherboards are used for System 2 Models 10 and 20.

USER CARD SLOTS	*TABLE MOUNT		19" RACK MOUNT		PANEL MOUNT		MOTHERBOARD	
	WITHOUT HARD DISK	WITH HARD DISK	WITHOUT HARD DISK	WITH HARD DISK	WITHOUT HARD DISK	WITH HARD DISK	TOTAL SLOTS	MODEL
*3	*BX6T-01	NA	NA	NA	BX6R-01	NA	*6	7112
6	BX9T-01	BX12T-02	NA	NA	BX9R-01	BX12R-02	9	7113
9	BX12T-01	BX15T-02	NA	NA	BX12R-01	BX15R-02	12	7114
12	BX15T-01	BX18T-02	NA	NA	BX15R-01	BX18R-02	15	7115
15	BX18T-01	BX21T-02	NA	NA	BX18R-01	BX21T-02	18	7116
18	BX21T-01	BX24T-02	NA	NA	BX21R-01	BX24R-02	21	7117
21	BX24T-01	NA	NA	NA	BX24R-01	NA	24	7118
23	BX26T-01	NA	NA	NA	BX26R-01	NA	26	7119
18	NA	NA	NA	BX26R-02	NA	NA	21	7117
23	NA	NA	BX26R-03	NA	NA	NA	26	7119

*Indicates default/standard configuration

Figure 10-1. Model 10 Card Racks

USER CARD SLOTS	*TABLE MOUNT		19" RACK MOUNT		PANEL MOUNT		MOTHERBOARD	
	WITHOUT HARD DISK	WITH HARD DISK	WITHOUT HARD DISK	WITH HARD DISK	WITHOUT HARD DISK	WITH HARD DISK	TOTAL SLOTS	MODEL
*6	*BX12T-02	BX15T-03	NA	NA	BX12R-02	BX15R-03	9	7113
9	BX15T-02	BX18T-03	NA	NA	BX15R-02	BX18R-03	12	7114
12	BX18T-02	BX21T-03	NA	NA	BX18R-02	BX21R-03	15	7115
15	BX21T-02	BX24T-03	NA	NA	BX21R-02	BX24T-03	18	7116
18	BX24T-02	NA	NA	NA	BX24R-02	NA	21	7117
15	NA	NA	NA	BX26R-04	NA	NA	18	7116
18	NA	NA	BX26R-03	NA	NA	NA	21	7117

*Indicates default/standard configuration

Figure 10-2. Model 20 Card Racks (with Internally Mounted 3.5-inch Floppy Disk Drive)

10.4.1 Card Rack Dimensions

Figures 10-3 through 10-6 provide the dimensions of each card rack. All drawings show the 20M-byte hard disk installed for reference. The internal 3.5-inch floppy drive in Model 20 is mounted in the same way. In the standard configuration of Model 10, the motherboard and card slots extend across the full length of the card rack.

Dimensions marked with an asterisk (*) are for mounting holes on the bottom of the card rack and M283 power supply. Mounting holes are 8-32 UNC threaded.

MODEL	A	B
BX06T-xx	4.71 (119.6)	3.95 (100.3)
BX09T-xx	6.58 (167.1)	5.82 (147.8)
BX12T-xx	8.46 (214.9)	7.70 (195.6)
BX15T-xx	10.33 (262.4)	9.57 (243.1)
BX18T-xx	12.21 (310.1)	11.45 (290.8)
BX21T-xx	14.08 (357.6)	13.32 (338.3)
BX24T-xx	15.96 (405.4)	15.20 (386.1)
BX26T-xx	17.21 (437.1)	16.45 (417.8)

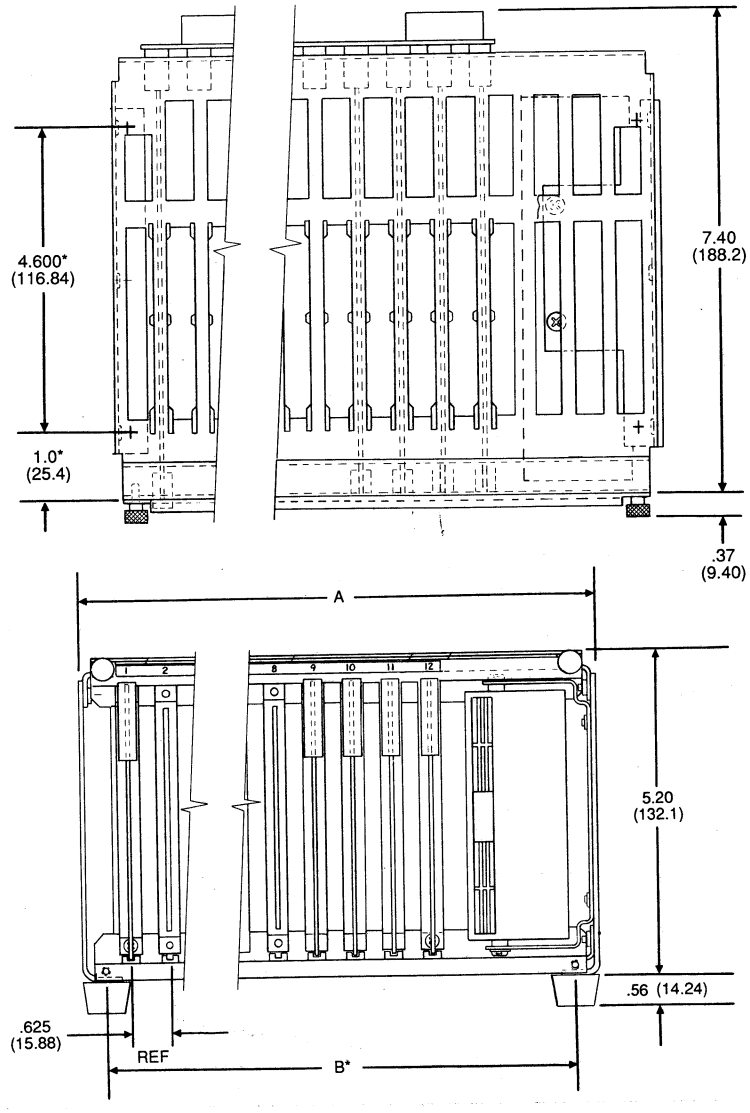


Figure 10-3. BXxx-T Card Rack Dimensions in Inches (mm)

MODEL	A	B
BX06R-xx	6.50 (165.1)	5.81 (147.6)
BX09R-xx	8.37 (212.6)	7.68 (195.1)
BX12R-xx	10.25 (260.4)	9.56 (242.8)
BX15R-xx	12.12 (307.8)	11.43 (290.3)
BX18R-xx	14.00 (355.6)	13.31 (338.1)
BX21R-xx	15.87 (403.1)	15.18 (385.6)
BX24R-xx	17.75 (450.9)	17.06 (433.3)
*BX26R-xx	19.00 (482.6)	18.31 (465.1)

*BX26R-02/03/04 19" rack mount versions have the rack mount brackets reversed from what is shown.

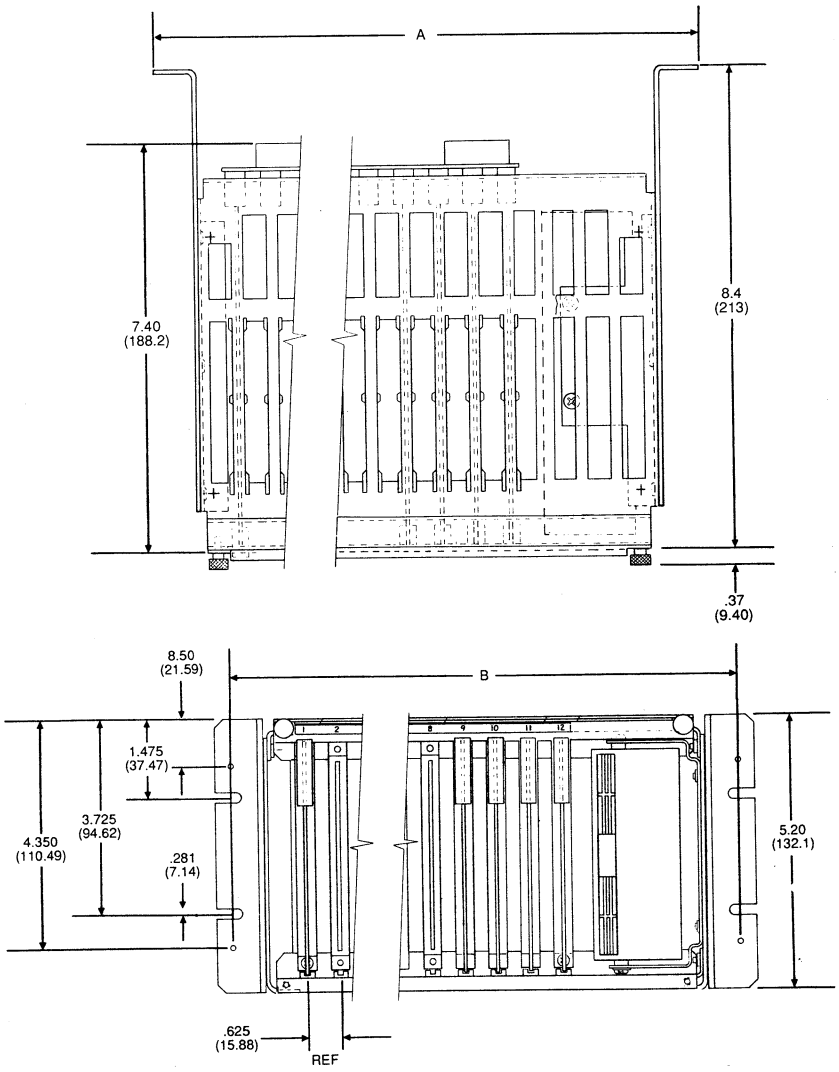


Figure 10-4. BXxx-R Card Rack Dimensions in Inches (mm)

MODEL	A	B
BX06T-xx w/M283	7.93 (201.8)	3.95 (100.3)
BX09T-xx w/M283	9.80 (249.3)	5.82 (147.8)
BX12T-xx w/M283	11.68 (297.0)	7.70 (195.6)
BX15T-xx w/M283	13.55 (344.6)	9.57 (243.1)
BX18T-xx w/M283	15.43 (392.4)	11.45 (290.8)
BX21T-xx w/M283	17.30 (439.9)	13.32 (338.3)
BX24T-xx w/M283	19.18 (487.6)	15.20 (386.1)
BX26T-xx w/M283	20.43 (519.3)	16.45 (417.8)

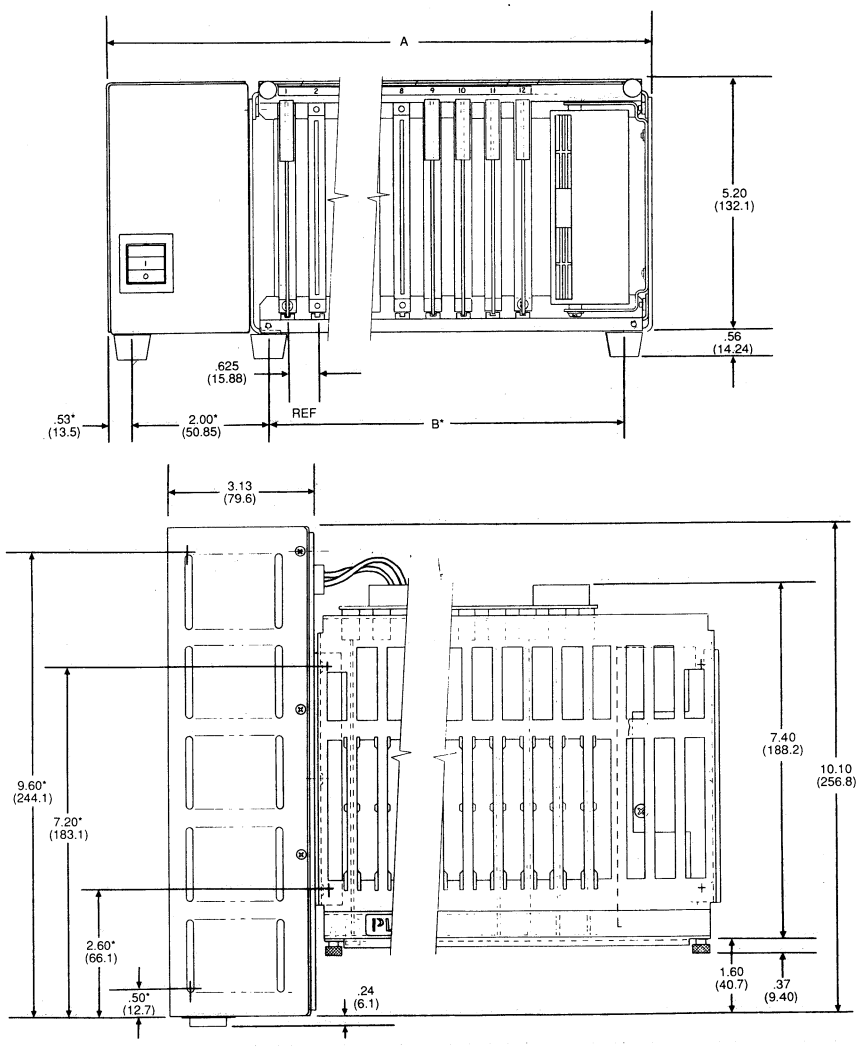


Figure 10-5. BXxx-T Card Rack Dimensions in Inches (mm) (with M283 Power Supply)

MODEL	A	B
BX06R-xx w/M283	9.72 (247.3)	9.03 (229.8)
BX09R-xx w/M283	11.59 (294.8)	10.9 (277.3)
BX12R-xx w/M283	13.47 (342.5)	12.78 (325.0)
BX15R-xx w/M283	15.34 (390.1)	14.65 (372.5)
BX18R-xx w/M283	17.22 (437.9)	16.53 (420.3)
BX21R-xx w/M283	19.09 (485.4)	18.40 (467.9)
BX24R-xx w/M283	20.97 (533.1)	20.28 (515.5)
BX26R-xx w/M283	22.22 (564.8)	21.53 (547.3)

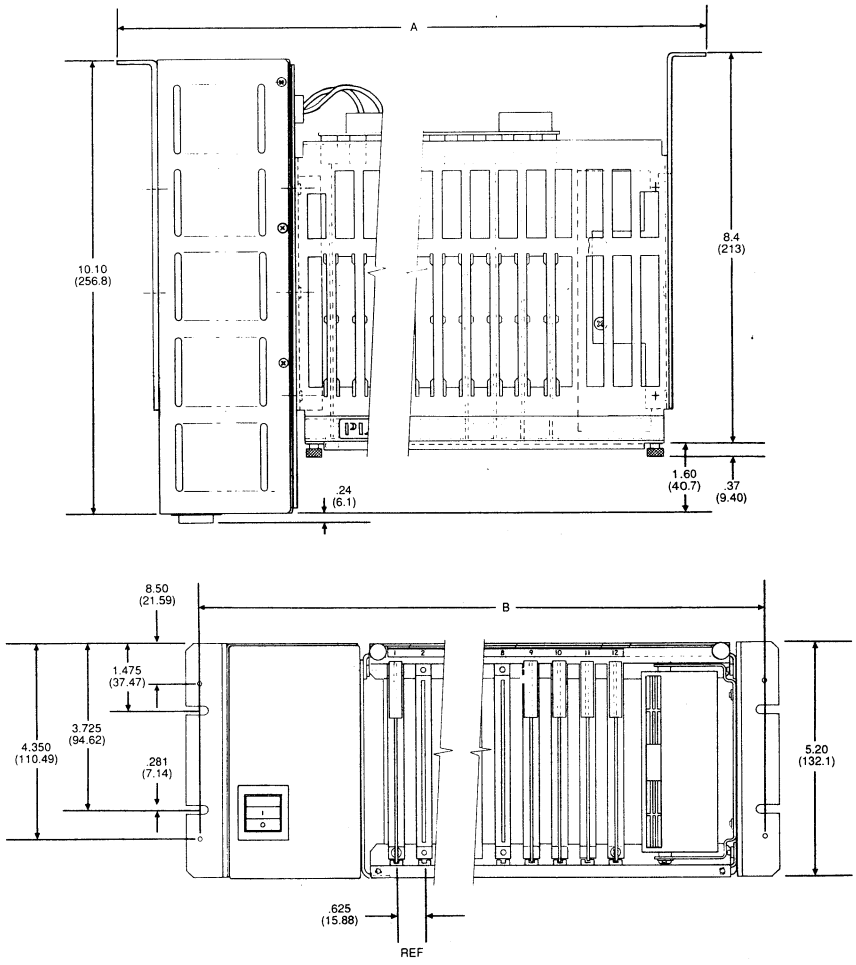


Figure 10-6. BXxx-R Card Rack Dimensions in Inches (mm) (with M283 Power Supply)

10.4.2 Power Connections

The 711X-Series motherboards are integral to the BX-Series card racks shipped with System 2. The 7112 motherboard provides one group of power connections, while the 7113 and 7114 motherboards provide two groups of power connections. For models 7115 and larger, there are three groups of power connections.

Terminal strips (Buchanan SSB4-series or equivalent) are provided for each group. The terminals are rated 15A and accept from 14 to 22 AWG wire. Figure 10-7 shows the power connections to the terminal strips on the back of the motherboard, as viewed from the back of the system. A 7114 was used for this illustration.

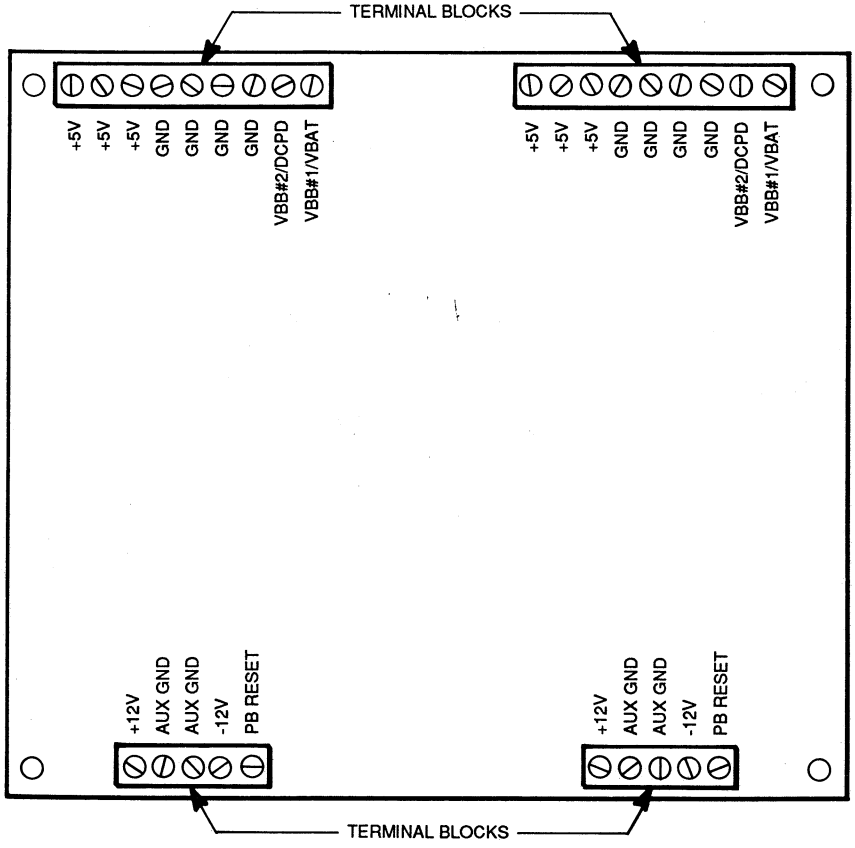


Figure 10-7. Power Connections

Chapter 11
Software Reference

The software supplied with System 2 can be divided into three parts: the BIOS (hardware drivers and initialization routines), system files (including MS-DOS and IO.SYS), and utilities. All software is the same for Model 10 and Model 20. The software schematic provided as a foldout at the end of this chapter illustrates the organization of the software resident in System 2.

11.1 BIOS FUNCTIONS

The BIOS program is included in the 64K ROM installed on the 7888-01/03 CPU card. It contains low-level hardware drivers and the code executed at power-on. The BIOS performs the following functions:

1. A limited power-on self-test (POST) that tests memory and all standard hardware present in the system. This code, in particular the memory tests performed, is simpler than in the IBM PC to ensure that the system is operating within five seconds after power is switched on.

2. Low-level initialization. This includes initialization of RAM data areas, interrupt vectors, and all hardware present in the system. RAM data areas in low memory from addresses 0:400 to 0:50F hex are PC-compatible. The BIOS includes initialization code for all standard hardware and support is included for BIOS extensions. These BIOS extensions are PROMs located on peripheral cards such as the 7391-01 floppy disk controller; the code in the PROM must perform the initialization and self-test functions for the associated hardware. BIOS extensions should be PC-compatible whenever possible.

3. Bootstrap loading. After BIOS initialization and self-test, this code searches for, loads, and executes the boot program contained in the boot sector of semiconductor, floppy, or hard disks. The boot program then loads and jumps to the MS-DOS operating system. The boot loader tries the floppy drive A, then the (optional) SCSI hard drive C, then the 7713 ROM Disk A, and finally the first optional memory disks (H-U), in that order. A failure to successfully find, load, or execute the boot program causes the loader to attempt to boot from the next drive in the sequence. (Pressing <Esc> at the console during the memory test causes the system to ignore drive C during this sequence. Note that <Esc> should only be pressed once.) If the operating system cannot be loaded from any disk, a failure message is displayed on the console and the system is halted. The default drive passed to the operating system after booting is the drive from which the system booted.

4. Hardware drivers. The BIOS contains the low-level drivers for all standard hardware in the system. In addition, drivers are included for certain optional hardware: four 7314A-1/7312-1 serial ports, up to four IBM-compatible 8250 serial ports (COMx), and up to four IBM-compatible parallel printer ports. Whenever possible, the software interrupts that provide the interface to these drivers are identical to those of the IBM PC. See Chapter 12 for a list of supported interrupts.

5. Additional PC-compatible BIOS services. Whenever possible, additional software interrupts used by the IBM PC BIOS are supported: for example, INT 15H, which returns the size of system memory. See Chapter 12 for a list of supported interrupts.

   The BIOS listing is included in Appendix A FOR REFERENCE ONLY.

   The BIOS routines should be accessed through software interrupts only. Applications should not depend on absolute addresses within the code segment.

11.2 SYSTEM FILES

The system files are MSDOS.SYS, IO.SYS, and COMMAND.COM, and are included either in ROM disk on the 7713 card or on a 3.5-inch disk. Backup copies of the system files are included on the MS-DOS VERSION 3.2 Disk. MSDOS.SYS (IBM PC-compatible version 3.2) and COMMAND.COM are used as supplied from Microsoft. Most of IO.SYS is written by Pro-Log and provides system-level initialization code and the interface between the MS-DOS operating system and the BIOS.

This interface between MS-DOS and the BIOS is the resident device drivers. The following character device drivers are included:

1. CON: This is the standard input/standard output device for operator interface. The actual physical console varies according to whether or not certain options are installed (see the discussion under *IO.SYS Initialization - Character Devices*, on page 11-4).

2. AUX: This is the standard auxiliary device, typically a serial port. Output to this device is usually directed to COM1, again depending on which options are installed (see the discussion under *IO.SYS Initialization - Character Devices*, on page 11-4).

3. PRN: This is the standard print device; output to it is sent to a 7392-01 IBM PC-compatible printer port, if one is present in the system. Otherwise, it is directed to SER2.

4. @CLOCK: This device is implemented with the 8254 Counter/Timer and the 8259 Interrupt Controller, exactly as in the IBM PC. The initial time and date known to the system is read from the battery-backed real-time clock during IO.SYS initialization.

5. COM1: This is a PC-compatible asynchronous RS-232-C communications port implemented with the 8250 UART on the 7888 CPU card. It may not be available for use in the system if no alternate optional console is installed in the system and the 7888's 8250 must be used as the console (see *IO.SYS Initialization - Character Devices* on page 11-4).

6. COM2-COM3: These are additional PC-compatible asynchronous RS-232-C communications ports. No STD BUS hardware is currently available to support these devices.

7. SER1-SER4: These are asynchronous RS-232-C communications devices that require the addition of the (optional) 7312-1 or 7314A-1 card to the core system. They are named SER in System 2 to distinguish them from COM ports, which are understood to be PC-hardware and BIOS-compatible communications ports. SER1 can be used as the console in certain configurations (see *IO.SYS Initialization - Character Devices* on page 11-4).

8. LPT1 - LPT3 - These are PC-hardware and BIOS-compatible printer ports. They require the addition of (optional) 7392-01 printer cards.

The resident block device drivers are:

1. ROM disk: This is the driver for the 7713 disks. The driver supports either one or two disks.

2. Rotating media: This driver supports up to three devices: the hard disk, which uses the 7390-02 SCSI card, and floppy disks that use the 7391-01 floppy disk card. BIOS extensions on the 7390-02 and 7391-01 provide the low-level drivers for this device driver.

3. Network drive: This is a dummy device driver that always returns an error in the present system.

4. Optional RAM/ROM disks: This driver supports up to 14 user-added 7714 ROM or 7715 RAM disks.

11.3 IO.SYS INITIALIZATION

11.3.1 Character Devices

The initialization code in IO.SYS reconfigures the resident device drivers if certain optional hardware is installed. The standard character driver configuration for System 2 with no options installed is:

CON:RS-232 port on CPU card (8250)
AUX: not usable, always returns an error
PRN: not usable, always returns an error
@CLOCK: generated by 8254 Counter/Timer
COM1: not usable, always returns an error
SER1-SER4: not usable, always returns an error
LPT1-LPT3: not usable, always returns an error

If an optional 7350 EGA/Keyboard card is installed, the character device driver configuration is:

CON: 7350 EGA/Keyboard card
AUX: RS-232 port on CPU card (8250)
PRN: not usable, always returns an error
@CLOCK: generated by 8254 Counter/Timer
COM1: RS-232 port on CPU card (8250)
SER1-SER4: not usable, always returns an error
LPT1-LPT3: not usable, always returns an error

If an optional 7312-1 or 7314A-1 is installed:

CON: RS-232 port Channel A on 7312-1 or 7314A-1
AUX: RS-232 port on CPU card (8250)
PRN: RS-232 port Channel B on 7312-1 or 7314A-1
@CLOCK: generated by 8254 Counter/Timer
COM1: RS-232 port on CPU card (8250)
SER1: not usable, always returns an error
SER2: RS-232 port Channel B on 7312-1 or 7314A-1
SER3-4: RS-232 ports Channels C-D on 7314A-1
LPT1-LPT3: not usable, always returns an error

If an optional 7350 EGA/Keyboard Card and an optional 7312-1 or 7314A-1 is installed:

CON: 7350 EGA/Keyboard card
AUX: RS-232 port on CPU card (8250)
PRN: RS-232 port Channel B on 7312-1 or 7314A-1
@CLOCK: generated by 8254 Counter/Timer
COM1: RS-232 port on CPU card (8250)
SER1-2: RS-232 ports Channels A-B on 7312-1 or 7314A-1
SER3-4: RS-232 ports Channels C-D on 7314A-1
LPT1-LPT3: not usable, always returns an error

If an optional 7392-01 printer interface card is installed (with any of the above configurations), the following devices are changed:

Printer (PRN): Printer port on 7392-01
LPT1: Printer port on 7392-01

Two additional 7392-01 printer interface cards can be added if their port address jumpers are changed (see the *7392 User's Guide*):

LPT2: Printer port on 7392-01 mapped at 378H
LPT3: Printer port on 7392-01 mapped at 278H

11.3.2 Block Devices

MODEL 10

The standard block driver configuration for the System 2 Model 10 with no options installed is:

Drive A: 7713 ROM disk A
Drive B: user-supplied 7713 ROM disk B, returns an error if not installed
Drive C: not usable, always returns an error
Drives D-F: not usable, reserved for future SCSI block devices, always return an error
Drive G: not usable, reserved for future network block devices, always returns an error
Drives H-U: not usable, always returns an error
Drives V and up: available for installable block device drivers

If you install the 7391-01 Floppy Disk Controller Card and one 3.5- or 5.25-inch floppy drive to the standard configuration of Model 10, the following drive letter assignment changes:

Drive B: First floppy drive (accessed by DS0)

If you purchased Model 10 with Option 020 (7713-02 installed), the following drive letter assignment is changed:

Drive B: 128K-byte battery-backed RAM disk

If you purchased Model 10 with Option 021 (7390-02 and 3.5-inch hard disk installed), the following drive letter assignment is changed:

Drive C: 20-megabyte hard disk

If you purchased Model 10 with Option 025 (7713-03 installed), the following drive letter assignments are changed:

Drive A: Unless you have installed a floppy drive, this device is not usable, always returns an error
Drive B: Unless you have installed two floppy drives, this device is not usable, always returns an error
Drive H: ROM disk (contains MS-DOS operating system)
Drive I: User-supplied 256K-byte ROM disk

If you purchased Model 10 with Option 026 (7713-04 installed), the following drive letter assignments are changed:

Drive A: Unless you have installed a floppy drive, this device is not usable, always returns an error
Drive B: Unless you have installed two floppy drives, this device is not usable, always returns an error
Drive H: ROM disk (contains MS-DOS operating system)
Drive I: 128K-byte battery-backed RAM disk

If optional 7714/15 memory disk cards are installed, the following drive letter assignments are changed:

Drives H-U: Each 7714 or 7715 card is addressed by MS-DOS as one memory disk drive within this range. The drive letter for each card is the letter selected on the rotary switch located on the front of the card. Each card must have a different drive letter selected. Drive letters H and I should not be selected when the 7713-03 or 7713-04 is installed.

MODEL 20

The standard block driver configuration for Model 20 with no options installed is:

Drive A: 3.5-inch floppy disk (accessed by DS0)
Drive B: not usable, always returns an error
Drive C: not usable, always returns an error
Drives D-F: not usable, reserved for future SCSI block devices, always return an error
Drive G: not usable, reserved for future network block devices, always returns an error
Drives H-U: not usable, always returns an error
Drives V and up: available for installable block device drivers

If you have installed an additional floppy drive (3.5- or 5.25-inch), the following drive assignment is changed:

Drive B: External floppy drive (accessed by DS1)

If you purchased Model 20 with Option 024 (7390-02 and 3.5-inch hard disk installed), the following drive letter assignment is changed:

Drive C: 20-megabyte hard disk.

If you purchased Model 20 with Option 025 (7713-03 installed), the following drive letter assignments are changed:

Drive H: ROM disk (contains MS-DOS operating system)
Drive I: User-supplied 256K-byte ROM disk

If you purchased Model 20 with Option 026 (7713-04 installed), the following drive letter assignments are changed:

Drive H: ROM disk (contains MS-DOS operating system)
Drive I: 128K-byte battery-backed RAM disk

If optional 7714/15 memory disk cards are installed, the following drive letter assignments are changed:

Drives H-U: Each 7714 or 7715 card is addressed by MS-DOS as one memory disk drive within this range. The drive letter for each card is the letter selected on the rotary switch located on the front of the card. Each card must have a different drive letter selected. Drive letters H and I

should not be selected when the 7713-03 or 7713-04 is installed.

11.4 SOFTWARE UTILITIES

We have included all utilities provided by Microsoft and additional utilities required to operate specific STD BUS hardware (such as MODE and FORMAT). We also provide terminal emulation, program transfer, and ROM disk generation utilities that run on the IBM PC.

All utilities are included on the STD-DOS Utilities Disk. In addition, as many Microsoft utilities and STD BUS hardware-specific utilities as possible are included on ROM disk A (or ROM Disk H) or the 3.5-inch MS-DOS VERSION 3.2 disk shipped with Model 20. Default AUTOEXEC.BAT and CONFIG.SYS files are also included on ROM disk A (or ROM Disk H).

11.4.1 Microsoft Utilities

The following Microsoft utilities are supplied:

ANSI.SYS	APPEND.EXE	ASSIGN.COM	ATTRIB.EXE
BACKUP.EXE	CHKDSK.EXE	COMMAND.COM	DEBUG.EXE
DISKCOMP	DISKCOPY	EDLIN.EXE	EXE2BIN.EXE
FC.COM	FIND.EXE	GRAFTABL.EXE	GRAPHICS.COM
JOIN.EXE	LABEL.EXE	LINK.EXE	MODE.COM
MORE.COM	PRINT.EXE	RAMDRIVE.SYS	RECOVER.EXE
REPLACE.EXE	RESTORE.EXE	SHARE.EXE	SORT.COM
SUBST.EXE	SYS.COM	TREE.EXE	XCOPY.EXE

11.4.2 Pro-Log Hardware-Specific Utilities

The following utilities are written by Pro-Log for use on the STD BUS:

SERMODE.COM: causes the PRN or AUX device I/O to be redirected to an asynchronous communications device (CON or SERx) on the 7312-1 or 7314A-1 card. It also sets options for and initializes the RS-232-C ports provided by the 7312-1/7314A-1 cards.

FORMAT.COM: formats hard and floppy disks that use the 7390 and 7391 cards.

HERC.EXE: configures the 7350-01 EGA/Keyboard Card to operate as a Hercules monographics adapter. It should be run before any Hercules-compatible application software is run.

@@XFER@@.EXE: provides the file transfer interface for the STD.EXE terminal emulation utility.

BREAKOFF.COM: disables break interrupt from 7312/14A

BREAKON.COM: enables break interrupt from 7312/14A

RTC.COM: sets the time and date in the system's real time clock.

XXMT.EXE and XRCV.EXE: used for Xmodem file transfer with CROSSTALK.

PARK.EXE: parks the head mechanism of the hard disk.

11.4.3 Terminal Emulation Utility

The STD.EXE terminal emulation utility allows the PC to act as the system console to System 2. It works with either the 7888 CPU Card or one of the 7312-1 or 7314A-1 Communications Cards as the console. Instructions for connecting a PC to System 2 and using the STD.EXE utility are provided in Chapter 3.

Support for CROSSTALK, an industry-standard terminal emulation program, is included with System 2. This support includes command and script files that are used to configure CROSSTALK on the PC and file transfer utilities (XXMT.EXE and XRCV.EXE) on ROM disk A (or ROM disk H) of System 2. The file transfer utilities are also included on the Pro-Log STD-DOS Utilities Disk. The CROSSTALK program itself is an option to the standard System 2.

The 7888 serial port must be configured as the console device when CROSSTALK is used. Therefore, the 7312-1, 7314A-1, and 7350 cards should not be installed.

11.4.4 Remote Operation of System 2

An installable device driver called TELE.SYS is included with System 2. When TELE.SYS is installed, System 2 uses the 7888 serial port connected to a modem as its terminal. A PC with its own modem and CROSSTALK (or other suitable software) can dial the System 2 and act as its console, making control of remote systems over telephone lines possible. A CROSSTALK command file (PLMODEM.XTK) is included to support this mode of operation.

XXMT.EXE and XRCV.EXE can be used in this configuration for file transfer to and from the remote system.

11.4.5 Utilities for Using COM1 to Transfer Files

XXMT.EXE and XRCV.EXE can also be used to transfer files between the STD system and a PC when the 7350-01 EGA/Keyboard card (or either 7312-1 or 7314A-1 card) is the console in System 2. A COM port on the PC is connected to the 7888 serial port (through the RC713 null modem); then one system runs XXMT.EXE and the other runs XRCV.EXE, and the file is transferred using the XMODEM protocol.

11.4.6 ROM Disk Generation and Transfer Utility

The ROMDISK.EXE utility can be used to format your application programs to a binary or Intel hex format so they can be downloaded to a PROM Programmer.

You will be asked for a list of files to be included on the new ROM disk. The output of the ROMDISK program is placed in files on the System 2 or PC's disks, in either binary or Intel hex format (your option). A file transmit utility named SENDFILE.EXE is also included to send these files to the programmer. Any programmer that can program 27512-type devices, has a serial port, and supports Intel hex or binary file transfer can be used with SENDFILE.EXE.

Chapter 8 contains information on how to run these utilities.

11.5 OPTIONAL SOFTWARE PACKAGES

The following software related to System 2 is available from Pro-Log:

CROSSTALK (from Microstuf): This is the terminal emulation utility used when the PC is the console for the STD-DOS system.

DoubleDOS (from Software Solutions): This PC memory-resident program allows two DOS applications programs to run concurrently.

STD-LIB: This is a library of software routines for Pro-Log distributed I/O cards.

Microsoft C, Macro assembler, and BASIC languages.

PAX (from Baker and Rabinowitz): This adds multitasking capability to MS-DOS.

LOOPWORKS (from Equinox): This program provides process monitoring and control without having to use a programming language.

11.6 DOCUMENTATION

11.6.1 Standard Documentation

This documentation is shipped with every System 2:

System 2 User's Guide (this document, Pro-Log model number 6001)
Hardware Reference binder (containing user's guides for the system cards)
DOS Reference ver. 3.2 (Pro-Log model number 6002)

11.6.2 Optional Documentation

This documentation is optional to System 2:

DOS Technical Reference (Pro-Log model number 6003)

Chapter 12
Technical Specifications

Unless otherwise noted, specifications apply to both Model 10 and Model 20. If installation of any option affects the specification, an alternate specification is given.

12.1 SPECIFICATIONS

12.1.1 General Specifications

Excluding mechanical disk drives:

Service life	10 years minimum
MTBF	12 years minimum
MTTR	5 minutes (excluding disk drives and card rack[1])
Warranty	5 years parts and labor

[1]Repair time is based on card swapping. Card rack and optional disk drive repair would require mechanical disassembly, with a MTTR of 30 minutes.

Winchester disk drive:

Service life	36,000 power-on hours; 10,000 start/stop cycles
MTBF	12,000 minimum power-on hours, excluding the first 50 hours
MTTR	30 minutes
Warranty	6 months

Floppy disk drive:

Service life	5 years; 15,000 power-on hours
MTBF	12,000 hours
MTTR	30 minutes
Warranty	6 months

FCC/UL Compliance

UL:	All System 2 components are UL-recognized.
FCC:	System 2 is exempt from FCC requirements for computing devices based on Part 15, Subpart J, Paragraph 15.801(c)(3), for industrial test equipment.

12.1.2 Environmental Specifications

MODEL 10

Operating:	
Free-Air Temperature Range	0° to +65°C
Noncondensing Relative Humidity	5 to 95%
Storage:	
Temperature Range	-40° to +70°C
Noncondensing Relative Humidity	5 to 95%

Model 10 with 3.5-inch hard disk drive option

Operating:	
Free-Air Temperature Range	+10° to +50°C
Noncondensing Relative Humidity	10 to 85%
Storage:	
Temperature Range	-25° to +70°C
Noncondensing Relative Humidity	5 to 90%

Model 20

Operating:	
Free-Air Temperature Range	+10° to +45°C
Noncondensing Relative Humidity	20 to 80%
Storage:	
Temperature Range	-10° to +60°C
Noncondensing Relative Humidity	5 to 90%

12.1.3 Mechanical Specifications

The STD BUS cards and card rack meet all general mechanical specifications for the *STD Series 7000 and 70C00 Specifications*, Pro-Log document number 111684. This document is included in the Pro-Log *Systems Data Book*, document number 112846.

SHOCK AND VIBRATION (OPERATING)

Model 10, no mechanical disk option:

Shock	45g peak acceleration, < 15 msec duration, square pulse
Vibration	3.0g peak acceleration, 10 to 100 Hz

Model 10, including hard disk option:

Shock	10.0g peak acceleration, < 10 msec duration, max. frequency 2/sec
Vibration	1.0g peak acceleration, 10 to 50 Hz

Model 20:

Shock	5.0g peak acceleration, < 10 msec duration, max. frequency 2/sec
Vibration	0.5g peak acceleration, 10 to 50 Hz

12.1.4 Electrical Specifications

1. All System 2 component cards meet the STDMG-8088 Bus Specification, Version 2.2

2. Power Requirements

PARAMETER	MIN	MAX	UNIT
DC Input Voltage, Vcc1	+4.85	+5.15	V
DC Input Voltage, AUX V+	+11.6	+12.4	V
DC Input Voltage, AUX V-	-11.6	-12.4	V
Model 10, no options:			
DC Input Current, +5V	--	0.7	A
DC Input Current, +12V	--	.05	A
DC Input Current, -12V	--	0.05	A
Model 20, no options:			
DC Input Current, +5V	--	1.3	A
DC Input Current, +12V	--	0.2	A
DC Input Current, -12V	--	0.05	A

1 All voltages 3% regulation, measured at any point on the STD backplane.

3. Real-time clock and battery-backed RAM data retention

	MIN	MAX	UNIT
Data retention time	40°C	10	yrs

4. Expansion slots

   System 2 card racks can contain up to 23 (Model 10) or 18 (Model 20) expansion slots for user-added STD BUS cards (depending on the factory-installed options specified). These expansion slots meet the STD BUS Electrical Specifications as defined in the *STDMG STD-8088 Bus Specification*, Pro-Log document number 106989, revision E and above. This document is included in the Pro-Log *Systems Data Book.* User-added cards must also meet the STDMG-8088 Bus Specification, to ensure compatibility with System 2's 7888 CPU card.

5. Memory, I/O, Interrupt, and RAM Data Area Maps

12.1.5 SYSTEM 2 MAPPING

Figures 12-1 through 12-4 are tables that show System 2 memory, I/O, and interrupt mapping. Each memory, I/O, and interrupt location is used in one of the following ways in System 2:

1. used by Pro-Log system hardware or software;
2. used by Pro-Log options;
3. used by MS-DOS software;
4. reserved by Pro-Log, Intel, IBM, or Microsoft for future use; or
5. available to the user for use in applications.

In general, these maps follow the standard set by the IBM PC wherever possible.

64K	FFFFF F0000	ROM BIOS (7888)	
64K	EFFFF E0000	ED000-EDFFF SCSI BUFFER EA000-EBFFF FLOPPY BUFFER E0000-E01FF MEMORY DISK	Pro-Log Reserved
128K	DFFFF C0000	DC000-DFFFF D0000-D7FFF P-L RESERVED CC000-CEFFF FLOPPY ROM C8000-CBFFF SCSI ROM C0000-C7FFF EGA ROM	ROM BIOS Extension
128K	BFFFF A0000	B8000-BFFFF CGA/EGA B0000-B3FFF MONOCHROME	Graphics Memory Display
512K	9FFFF 20000	(ON STD BUS)	System RAM
128K	1FFFF 00000	(ON 7888)	

Figure 12-1. System 2 Memory Map

1. PC-DOS COMPATIBLE ROM BIOS DATA AREA:

0400	DWx4	RS232_BASE	address of RS232 adapters
0408	DWx4	PRINTER_BASE	address of printers
0410	DW	EQUIP_FLAG	installed hardware code
0412	DB	INIT_FLAG	initialization flag
0413	DW	MEM_SIZE	memory size in kbytes
0415	DB	MFG_ERR_FLAG	mailbox for mfg test error codes
0416	DB		?
	KEYBOARD data areas:		
0417	DB	KBD_FLAG	1st byte of keyboard status
0418	DB	KBD_FLAG_1	2nd byte of keyboard status
0419	DB	ALT_INPUT	storage for alternate kbd input
041A	DW	KBD_BUF-HEAD	ptr to head of kbd bufr
041C	DW	KBD_BUF_TAIL	ptr to tail of kbd bufr
041E	DWx16	KBD_BUFFER	bufr for up to 15 entries
	DISKETTE data areas:		
043E	DB	SEEK_STATUS	drive recalibration status
043F	DB	MOTOR_STATUS	motor status flag
0440	DB	MOTOR_COUNT	time out counter
0441	DB	DISK_STATUS	return code status byte
0442	DBx7	NEC_STATUS	status byte from NEC
	VIDEO display data areas:		
0449	DB	CRT_MODE	current CRT mode
044A	DW	CRT_COLS	number of columns on screen
044C	DW	CRT_LENGTH	length of regen in bytes
044E	DW	CRT_START	start addr in regen bufr
0450	DWx8	CURS_POSITION	cursor pos for up to 8 pages
0460	DW	CURS_MODE	current curs mode code
0462	DB	ACTIVE_PAGE	current page on screen
0463	DW	ADDR_6845	base addr for disp card
0465	DB	CRT_MODE_SET	cur code of 3 x 8 register
0466	DB	CRT_PALETTE	current crt color card
	POST data areas:		
0467	DW	IO_ROM_INIT	ptr to (opt) i/o rom init routine
0469	DW	IO_ROM_SEG	ptr to rom segment
046B	DB	INTER_FLAG	interrupt occured flag
	TIMER data areas:		
046C	DW	TIMER_LOW	low word of timer count
046E	DW	TIMER_HIGH	high word of timer count
0470	DB	TIMER_OFLW	timer rollover flag
	SYSTEM data areas:		
0471	DB	BIOS_BREAK	bit 7 = 1 if break key hit
0472	DW	RESET_FLAG	word = 1234 if kbd reset underway

Figure 12-2. System 2 RAM Data Areas,Segment 0000

```
                FIXED DISK data areas:
0474    DW      FLP_INT_ADR
0476    DW      FLP_INT_SEG

                PRINT & RS232 timeout variables:
0478    DBx4    PRNT_TIMEOUT
047C    DBx4    RS232_TIMEOUT

                more KEYBOARD data areas:
0480    DW      BUFF_START
0482    DW      BUFF_END

0484    DBx7                            unused ram

                more FLOPPY data areas:
048B    DW      RES_BIOS_FLAG

                more HARD DISK data areas:
048D    DW      DISK_DATA_SEG
048F    DB      HF_INT_FLAG
0490    DB      HF_CNTL

                more DISKETTE data areas:
0491    DB      DISK_STATE
0492    DB
0493    DB
0494    DB
0495    DB      DISK_TRK
0496    DB
0497    DB

                KEYBOARD led flag:
0498    DB      KBD_FLAG_2

                RTC data areas:
0499    DW      RTC_USER_ADDR
049B    DW      RTC_USER-SEG
049D    DW      RCT_LOWCOUNT
049F    DW      RTC_HICOUNT
04A1    DB      RTC_WAIT

                Unused RAM data area:
04A2    DBx5E

0500    DB      STATUS_BYTE

                Reserved (DOS and BASIC) RAM data area:
0501    DBx31
```

Figure 12-2. System 2 RAM Data Areas, Segment 0000 (Continued)

2. STD-DOS ROM BIOS DATA AREA:

0520	DBx15	SER1_DATA	7314/12 mailboxes
052F	DBx15	SER2_DATA	
053E	DBx15	SER3_DATA	
054D	DBx15	SER4_DATA	
055C	DBx32	BUF1	32-byte (16 character)
057C	DBx32	BUF2	input buffers for SCC
059C	DBx32	BUF3	character devices
05BC	DBx32	BUF4	
05DC	DWx64	DRVSTACK	temporary device driver stack
065C	Wx32	INSTACK	temporary hdw interrupt stack
069C	DW	STACK_SP	temp SP storage for device drivers
069E	DW	STACK_SS	temp SS storage for device drivers
06A0	DW	INTR_SP	temp SP storage for hdw interrupts
06A2	DW	INTR_SS	temp SS storage for hdw interrupts
06A4	DD	PRTRSAVE	request block pointer

Miscellaneous STD-DOS mailboxes:

06AB	DB	COM1CHAR	
06A9	DB	COM1READY	
06AA	DB	INNULSEQ	
06AB	DB	BRKFLAG	
06AC	DB	SCANCODE	
06AD	DB	SCCFLAG	
06AE	DB	CON_RESET_ENABLE	enables ESC c reset from CON
06AF	DB	DRVCOUNT	this mailbox used by DSKDRVR
06B0	DW	DAYSCOUNTED	for use with clock routines
06B2	DB	RAMDRVBFLAG	these mailboxes used by
06B3	DW	MEMDSKMASK	device drivers
06B5	DB	ESC_FLAG	
06B6	DW	CLS_PTR	

Figure 12-2. System 2 RAM Data Areas, Segment 0000 (Continued)

PORTS	USAGE
000-01F	*DMA controller #1(8237A)
020-03F	***Interrupt controller** #1 (82C59A)
040-05F	***Counter/timer** (82C54)
060-06F	***PPI** (8255A-5, **selected functions only**)
070-07F	*Real-time clock, AT-compatible
080-09F	*DMA page registers
0A0-**0AF**	**NMI port**
0B0-0BF	*Interrupt controller #2
0C0-0DF	*DMA controller #2
0E0-0EF	IBM RESERVED
0F0-0FF	*Math Coprocessor
100-1EF	Customer use
1F0-1FF	*Fixed disk (AT)
200-20F	*Game control
210-21F	*Expansion unit
220-24F	*IBM RESERVED
250-26F	PRO-LOG RESERVED
270-27F	***Printer #2** (7392 printer option)
280-29F	**Memory disks** (7714/15 memory disk options)
2A0-2BF	PRO-LOG RESERVED
2C0-2CF	**SCSI card** (7390 SCSI option)
2D0-2DF	**Floppy controller card** (7391 FDC option)
2E0-2EF	PRO-LOG RESERVED
2F0-2FF	IBM RESERVED & Async comm #2
300-31F	Prototype cards
320-32F	*Fixed disk (XT)
330-35F	PRO-LOG RESERVED
360-36F	*IBM RESERVED
370-37F	***Printer #1** (7392 printer option)
380-38F	*SDLC/BI-SYNC comm #2
390-39F	Customer use
3A0-3AF	*BI-SYNC comm #2
3B0-3BF	***Monochrome display/printer** (7350/7392)
3C0-3CF	*IBM RESERVED
3D0-3DF	***Color graphics display** (7350 EGA option)
3E0-3EF	*IBM RESERVED
3F0-3FF	*Diskette and **Async comm #1**
400-FFFF	(not used)

*IBM-compatible hardware
Boldface indicates I/O ports used in System 2

Figure 12-3. STD-DOS I/O Map, System 2

INTERRUPT	USAGE
00	**Divide by 0** (dedicated hdw interrupt)
01	**Single step** (dedicated hdw interrupt)
02	**NMI** (dedicated hdw interrupt)
03	**Breakpoint** (dedicated hdw interrupt)
04	**Overflow** (dedicated hdw interrupt)
05	***Print screen**
06-07	IBM RESERVED
08	***System timer hardware interrupt**
09	***Keyboard hardware interrupt** (7350 EGA option)
0A	***CRT vertical retrace** (7350 EGA option)
0B	**Customer hardware interrupt** (front of 7888)
0C	***Communications hardware interrupt**
0D	**Real-time clock hardware interrupt**
0E	**VBAT low hardware interrupt**
0F	**STD BUS INTRQ**
10	***Video display BIOS service** (7350 EGA option)
11	***Equipment check BIOS service**
12	***Memory size BIOS service**
13	***Diskette BIOS service** (7391 FDC option)
14	***Communications BIOS service**
15	*Cassette
16	***Keyboard BIOS service** (7350 EGA option)
17	***Printer BIOS service** (7392 printer option)
18	Resident BASIC
19	***Bootstrap**
1A	***Time and date BIOS service**
1B	***Keyboard break**
1C	***Timer tick**
1D	***Video initialization** (7350 EGA option)
1E	***Diskette parameters** (7391 FDC option)
1F	***Video graphics** (7350 EGA option)
20-3F	**MS-DOS RESERVED**
40-5F	IBM RESERVED
60-67	Customer use
68-70	PRO-LOG RESERVED
71	**SERn software interrupt**
72	PRO-LOG RESERVED
73	**Clock software interrupt**
74-76	PRO-LOG RESERVED
77	**Memdisk I/O BIOS service**
78	**Mechanical drive software int** (7390/7391)
79	**SCSI software interrupt** (7390 SCSI option)
7A-7F	PRO-LOG RESERVED
80-9F	**SERn hardware interrupts** (7312/14 option)
A0-A3	PRO-LOG RESERVED
A4	**SCSI hardware interrupt** (7390 SCSI option)
A5	**Floppy hardware interrupt** (7391 FDC option)
A6-BF	PRO-LOG RESERVED
C0-FF	Customer use

*IBM-compatible interrupt routine
Boldface indicates interrupts used in System 2

Figure 12-4. STD-DOS Interrupt Map, System 2

Chapter 13

Maintenance

13.1 REFERENCE DRAWINGS

The schematic and assembly drawings for the system cards are provided in the user's guides included in the Hardware Reference binder FOR REFERENCE USE ONLY. These drawings may differ in some respects from the card you receive from Pro-Log. If you require the specific revision documents of the schematic and assembly for the board shipped, please call Pro-Log Customer Support at (800) 538-9570 from 7 a.m. to 5 p.m. (Pacific Time).

13.2 RETURN FOR REPAIR/WARRANTY

Return for Repair Procedures

Domestic Customers:

1. Call Customer Support at (800) 538-9570 from 7 a.m. to 5 p.m. (Pacific Time).

2. Explain the problem and we may be able to solve it on the phone. If not, we will give you a Customer Return Order (CRO) number.

3. Please be sure to enclose a packing slip with CRO number, serial number of the equipment, if applicable, and reason for the return, with the name and telephone number of the person we should contact (preferably the user), if we have any further questions.

4. Pack the equipment in a solid cardboard box secured with packing material.

CAUTION

Loose MOS integrated circuits, or any product containing CMOS integrated circuits, must be protected from electrostatic discharge during handling and shipping. Use conductive foam pads or conductive plastic bags, and never place MOS or CMOS circuitry in contact with Styrofoam materials.

5. Ship prepaid and insured to:

   Pro-Log Corporation
   2560 Garden Road
   Monterey, California 93940

   Reference CRO #_____

International Customers

Equipment repair is handled by your Pro-Log distributor. If you need to contact Pro-Log, the factory can be reached at any time by TWX at 910-360-7082.

Limited Warranty

Term: Five years parts and labor warranty from date of shipment for STD BUS hardware.[1]

Limited Warranty: Seller warrants that the articles furnished hereunder are free from defects in material and workmanship and perform to applicable, published Pro-Log specifications. This warranty is in lieu of any other warranty expressed or implied. In no event will Seller be liable for special or consequential damages as a result of any alleged breach of this warranty provision. The liability of Seller hereunder shall be limited to replacing or repairing, at its option, any defective units which are returned F.O.B. Seller's plant. Equipment or parts which have been subject to abuse, misuse, accident, alteration, neglect, unauthorized repair or installation are not covered by warranty. Seller shall have the right of final determination as to the existence and cause of defect. As to items repaired or replaced, the warranty shall continue in effect for the remainder of the warranty period, or for ninety (90) days following date of shipment by seller of the repaired or replaced part whichever period is longer. No liability is assumed for expendable items such as lamps and fuses. No warranty is made with respect to custom equipment or products produced to Buyer's specifications except as specifically stated in writing by Seller and contained in the contract.

[1] Power Supplies, mechanical disk drives, and cables excepted. Disk drives are warranted for six months; power supplies are warranted for one year; and cables are warranted for five years.

Appendix A
BIOS Listing

```
                                PAGE    80,132
                                TITLE   ROM BIOS for STD-DOS 2.00
                        ;*****************************************************************************
                        ;*  (C)Copyright 1985,1986, 1987 Pro-Log Corp.                               *
                        ;*****************************************************************************
                        C               INCLUDE DRVDCL.INC
                        C               SUBTTL  Device Driver Declarations
                        C               PAGE 80,132
                        C ;*****************************************************************************
                        C ;*  Device Driver Declarations for BIOS and IO.SYS                           *
                        C ;*****************************************************************************
                        C ;
                        C                                       ;Attribute word format:
                        C DEVATTRIB     RECORD  CHR:1,IOCTL:1,IBM:1,DMY:1,OPEN:1,DMY2:4, VER32:1,DMY3:1,SPC:1,CLK:1,NIL:1,STO:1
                          ,STI:1
                        C
= 0000                  C WRITE_PROTECT       EQU  0                    ;Device Driver Error Codes and Status Bits
= 0001                  C UNKNOWN_UNIT        EQU  1                    ;
= 0002                  C NOT_READY           EQU  2                    ;
= 0003                  C UNKNOWN_COMMAND     EQU  3                    ;
= 0004                  C CRC_ERROR           EQU  4                    ;
= 0005                  C BAD_LENGTH          EQU  5                    ;
= 0006                  C SEEK_ERROR          EQU  6                    ;
= 0007                  C UNKNOWN_MEDIA       EQU  7                    ;
= 0008                  C SECTOR_NOT_FOUND    EQU  8                    ;
= 0009                  C OUT_OF_PAPER        EQU  9                    ;
= 000A                  C WRITE_FAULT         EQU  10                   ;
= 000B                  C READ_FAULT          EQU  11                   ;
= 000C                  C GENERAL_FAILURE     EQU  12                   ;
                        C ;RESERVED           EQU  13                   ;
                        C ;RESERVED           EQU  14                   ;
= 000F                  C INVALID_DISK_CHANGE EQU  15                   ;
                        C
                        C STATREC           RECORD  ERR:1,STATRES:5,BUSY:1,REQDONE:1,ERROR:8  ;status word format
                        C
= 0011                  C MAX_COMMAND       EQU   17                    ;number of functions for a device driver
                        C
                        C STATICHDR         STRUC                       ;Static Request Header
0000  ??                C REQUESTLENGTH     DB    ?                     ;this structure shows the format of the
0001  ??                C UNIT              DB    ?                     ;   static header on all request blocks.
0002  ??                C COMMAND           DB    ?                     ;
0003  ????              C STATUS            DW    ?                     ;
0005  0002[             C LINKS             DD    2 DUP (?)             ;
          ????????      C
                    ]   C
                        C
000D                    C STATICHDR         ENDS                        ;
                        C
                        C RWSTRUC           STRUC                       ;this structure shows the format of a read,
0000  000D[             C                   DB    (SIZE STATICHDR) DUP (?);write or write with verify request.
          ??            C
                    ]   C
                        C
000D  ??                C MEDIADESCRIPTOR   DB    ?                     ;
000E  ????????          C XFERADDRESS       DD    ?                     ;
0012  ????              C XFERCOUNT         DW    ?                     ;
0014  ????              C FIRSTSECTOR       DW    ?                     ;
0016  ????????          C VOLUMEIDONERROR   DD    ?                     ;
001A                    C RWSTRUC           ENDS                        ;
                        C
                        C RNDSTRUC          STRUC                       ;this structure shows the format of a
0000  000D[             C                   DB    (SIZE STATICHDR) DUP (?);   read non-destructive request.
          ??            C
                    ]   C
                        C
000D  ??                C NEXTBYTE          DB    ?                     ;
000E                    C RNDSTRUC          ENDS                        ;
                        C
                        C MCSTRUC           STRUC                       ;this structure shows the format of a
0000  000D[             C                   DB    (SIZE STATICHDR) DUP (?);   media check request.
          ??            C
                    ]   C
                        C
000D  ??                C CURRENTMEDIA      DB    ?                     ;
000E  ??                C MEDIASTATUS       DB    ?                     ;
000F  ????????          C PREVOLUMEIDPTR    DD    ?                     ;
0013                    C MCSTRUC           ENDS                        ;
```

```
                                C
                                C BUILDSTRUC        STRUC                       ;this structure shows the format
0000  000D[                     C                   DB    (SIZE STATICHDR) DUP (?);   of a build BPB request.
        ??                      C
                        ]       C
                                C
000D  ??                        C MEDIAINFAT        DB    ?                     ;
000E  ????????                  C SCRATCHADDRESS    DD    ?                     ;
0012  ????????                  C BPBADDRESS        DD    ?                     ;
0016                            C BUILDSTRUC        ENDS                        ;
                                C
                                C INITSTRUC         STRUC                       ;this structure shows the format
0000  000D[                     C                   DB    (SIZE STATICHDR) DUP (?);   of an Init request.
        ??                      C
                        ]       C
                                C
000D  ??                        C NUMBEROFUNITS     DB    ?                     ;
000E  ????????                  C BREAKADDRESS      DD    ?                     ;
0012  ????????                  C BPBPOINTER        DD    ?                     ;
0016  ??                        C DEVICENUMBER      DB    ?                     ;
0017                            C INITSTRUC         ENDS                        ;
                                C
                                C GENIOCTLSTRC      STRUC
0000  000D[                     C                   DB    (SIZE STATICHDR) DUP (?)
        ??                      C
                        ]       C
                                C
000D  ??                        C CATEGORY          DB    ?
000E  ??                        C IOCTLFUNCTION     DB    ?
000F  ????                      C SICONTENTS        DW    ?
0011  ????                      C DICONTENTS        DW    ?
0013  ????????                  C IOCTLPARAM        DD    ?
0017                            C GENIOCTLSTRC      ENDS
                                C
                                C ;     ******    GENERIC IOCTL GET/SET DEVICE PARAMETERS     ******
                                C
                                C IOCTLDEVPAR       STRUC
0000  ??                        C SFUNCTIONS        DB    ?
0001  ??                        C DEVICETYPE        DB    ?
0002  ????                      C DEVICEATTRIB      DW    ?
0004  ????                      C NUMCYLINDERS      DW    ?
0006  ??                        C MEDIATYPE         DB    ?
0007  0013[                     C CUR_BPB           DB    (SIZE BPB) DUP (?)
        ??                      C
                        ]       C
                                C
001A  0010[                     C TRACKLAYOUT       DB    16 DUP (?)
        ??                      C
                        ]       C
                                C
002A                            C IOCTLDEVPAR       ENDS
                                C
                                C ;     ******    GEN IOCTL READ/WRITE/FMT TRACK PARM BLOCK     ******
                                C
                                C RD_WR_FMT_TRK     STRUC
0000  ??                        C SFUNCTION         DB    ?
0001  ????                      C HEAD              DW    ?
0003  ????                      C CYLINDER          DW    ?
0005  ????                      C STARTSECTOR       DW    ?
0007  ????                      C NUMOFSECTORS      DW    ?
0009  ????????                  C TRANSFERADR       DD    ?
000D                            C RD_WR_FMT_TRK     ENDS
                                C
                                C
                                C BPB               STRUC                       ;BPB format
0000  ????                      C BYTESPERSECTOR    DW    ?                     ;this structure shows the format of a BPB
0002  ??                        C SECTORSPERCLUSTER DB    ?                     ;
0003  ????                      C RESERVEDSECTORS   DW    ?                     ;
0005  ??                        C NUMBEROFFATS      DB    ?                     ;
0006  ????                      C NUMBERDIRENTRIES  DW    ?                     ;
0008  ????                      C TOTALSECTORS      DW    ?                     ;
000A  ??                        C MEDIABYTE         DB    ?                     ;
000B  ????                      C SECTORSPERFAT     DW    ?                     ;
000D  ????                      C SECTORSPERTRACK   DW    ?
000F  ????                      C HEADS             DW    ?
0011  ????                      C HIDDENSECTORS     DW    ?
```

```
0013                    C  BPB              ENDS                        ;
                        C
                        C  DIRECTORYENTRY   STRUC                       ;this structure shows the organization
0000  0008[             C  FILENAME         DB    8 DUP (?)             ;   of an entry in an MSDOS directory.
        ??              C
                    ]   C
                        C
0008  0003[             C  FILENAMEEXT      DB    3 DUP (?)             ;
        ??              C
                    ]   C
                        C
000B  ??                C  FILEATTRIBUTE    DB    ?                     ;
000C  000A[             C  FILERESERVED     DB    10 DUP (?)            ;
        ??              C
                    ]   C
                        C
0016  ????              C  FILETIME         DW    ?                     ;
0018  ????              C  FILEDATE         DW    ?                     ;
001A  ????              C  FILECLUSTER1     DW    ?                     ;
001C  ????              C  FILESIZELOW      DW    ?                     ;
001E  ????              C  FILESIZEHIGH     DW    ?                     ;
0020                    C  DIRECTORYENTRY   ENDS                        ;
                        C
                        C  LINKLIST         STRUC
0000  ????              C  NEXTDEVICE       DW    ?
0002  ????              C  NXTDEVSEG        DW    ?
0004  ????              C  DEVATTRIBUTES    DW    ?
0006  ????              C  DEVSTRATEGY      DW    ?
0008  ????              C  DEVINTERRUPT     DW    ?
000A  0008[             C  DEVNAME          DB    8 DUP (?)
        ??              C
                    ]   C
                        C
0012                    C  LINKLIST         ENDS
                        C
                        C  BOOT_SECTOR      STRUC                       ;
0000  0003[             C  START            DB      3 DUP (?)           ;
        ??              C
                    ]   C
                        C
0003  0008[             C  OEM_NAME         DB      8 DUP (?)           ;
        ??              C
                    ]   C
                        C
000B  0013[             C  DISKBPB DB       (SIZE BPB) DUP (?)
        ??              C
                    ]   C
                        C
001E                    C  BOOT_SECTOR      ENDS                        ;
                        C
= 0040                  C  SET_DEV_CODE     EQU   40H
= 0060                  C  GET_DEV_CODE     EQU   60H
= 0041                  C  WRITE_TRK_CODE   EQU   41H
= 0061                  C  READ_TRK_CODE    EQU   61H
= 0042                  C  FORMAT_TRK_CODE  EQU   42H
= 0062                  C  VER_TRK_CODE     EQU   62H
                        C
= 0284                  C  RAMBASEPORT      EQU     0284H
= 0280                  C  ROMBASEPORT      EQU     0280H
                        C
= E000                  C  DISKSEG          EQU     0E000H              ;segment addr of memory disks
= 0001                  C  BAD_CMD          EQU     001H                ;returned status
= 0003                  C  WRITE_FAIL       EQU     003H                ;
= 0077                  C  MDISK_TRAP       EQU     77H
                        C                                               ;SCC char device declarations:
= 0071                  C  SCC_TRAP             EQU   71H               ;SCC BIOS trap
                        C
= 0000                  C  SCC_INIT         EQU   0                     ;SCC_TRAP functions
= 0001                  C  SCC_SEND         EQU   1                     ;
= 0002                  C  SCC_BUF_READ     EQU   2                     ;
= 0003                  C  SCC_PORT_STAT    EQU   3                     ;
= 0004                  C  SCC_BUF_STAT     EQU   4                     ;
                        C
= 0000                  C  LPT_SEND         EQU     0
= 0001                  C  LPT_INIT         EQU     1
= 0002                  C  LPT_STAT         EQU     2
```

```
= 0017                  C  LPT_TRAP      EQU     17H
                        C
= 000E                  C  VIDEO_TTY     EQU     0EH
= 0010                  C  VIDEO_INT     EQU     10H
= 0000                  C  VIDEO_MODE    EQU     0
= 000F                  C  VIDEO_STATE   EQU     15
                        C
                        C  SCCIN         MACRO                   ;input & delay 8 clocks for
                        C                IN      AL,DX           ;SCC recovery
                        C                XCHG    DX,DX           ;
                        C                XCHG    DX,DX           ;
                        C                ENDM                    ;
                        C
                        C  SCCOUT        MACRO                   ;output & delay 8 clocks for
                        C                OUT     DX,AL           ;SCC recovery
                        C                XCHG    DX,DX           ;
                        C                XCHG    DX,DX           ;
                        C                ENDM                    ;
                        C
= 001B                  C  ESC               EQU   1BH           ;characters
= 000D                  C  CR                EQU   0DH           ;
= 000A                  C  LF                EQU   0AH           ;
= 0007                  C  BELL              EQU   07H           ;
= 0003                  C  ETX               EQU   03H           ;^C
= 0024                  C  MSGEND            EQU   '$'           ;
                        C
                        C  ;------The following equates are used by EGAKYBD.ASM
= 0000                  C  FALSE             EQU   0
= FFFF                  C  TRUE              EQU   NOT FALSE
= 0003                  C  CTLC              EQU   03H
= 0010                  C  CTLP              EQU   10H
= 0000                  C  NUL               EQU   00H
= 0072                  C  CTL_PRNSCR        EQU   0072H
= 0000                  C  CTL_BREAK         EQU   0000H
= 0016                  C  KYBD_TRAP         EQU   16H
= 0029                  C  TTY_TRAP          EQU   29H
= 0001                  C  KYBD_CHECK        EQU   01H
= 0000                  C  KYBD_READ         EQU   00H
= 0061                  C  KYBD_PORT         EQU   61H
= 00FF                  C  FFPORT            EQU   0FFH
                        C
                        C  ;------The following equates are used by CONDRIV.ASM
= 0069                  C  XMIT_@@CON        EQU   69H
= 0003                  C  CHECK_COM1        EQU   03H
= 0002                  C  RECV_COM1         EQU   02H
= 0001                  C  XMIT_COM1         EQU   01H
= 0000                  C  INIT_COM1         EQU   00H
= 0001                  C  COM1DATRDY        EQU   01H
= 0020                  C  COM1XM1TRDY       EQU   20H
= 0080                  C  COM1TIMEOUT       EQU   80H
= 0014                  C  COM1_TRAP         EQU   14H
= 03F8                  C  BASE_8250         EQU   03F8H
= 0201                  C  GAMEPORT          EQU   0201H
= 0062                  C  COPRPORT          EQU   0062H
= 0062                  C  COPRID            EQU   62H
                        C
= 0080                  C  VECT14A           EQU   80H           ;   7314 SCC1, channels A and B
= 0081                  C  VECT14C           EQU   81H           ;   7314 SCC2, channels C and D
                        C
= 00B1                  C  CHA14             EQU   00B1H         ;
= 00B3                  C  CHB14             EQU   00B3H         ;
= 00B5                  C  CHC14             EQU   00B5H         ;
= 00B7                  C  CHD14             EQU   00B7H         ;
                        C
= 0018                  C  MAXHOURSMS        EQU     0018H       ;24 hours max count
= 00B0                  C  MAXHOURSLS        EQU     00B0H       ;
= 001C                  C  USERINT           EQU     1CH         ;user prog interupt
                        C
= AA55                  C  EXTND_BIOS_ID     EQU     0AA55H      ;These equates are for search
= C000                  C  VIDEOSTART        EQU     0C000H      ;and installing bios extensions
= C800                  C  VIDEOEND          EQU     0C800H      ;
= C800                  C  BIOSTART          EQU     0C800H      ;
= E000                  C  BIOSEND           EQU     0E000H      ;
= 0005                  C  HARD_DISK           EQU     05H
= CC00                  C  FLOPPYSEG         EQU     0CC00H      ;constants for boot loader
= C800                  C  SCSISEG           EQU     0C800H      ;
```

```
= 0010                  C  FLOPPYFLAG        EQU   10H                 ;
= 0004                  C  SCSIFLAG          EQU   04H                 ;
= 0006                  C  BOOTOFFSET        EQU   0006H               ;
                        C
                        C  ;------The following port assignments are for PC compatibility
= 0020                  C  INTEND            EQU   20H                 ;end of interupt code
= 0020                  C  PORTCNTL_8259     EQU   0020H               ;8259 control port
= 0021                  C  PORTDATA_8259     EQU   0021H               ;8259 data port
= 00FE                  C  INTERUPT_MASK     EQU   00FEH               ;mask int 8 only
                        C
= 0000                  C  VAL_GET_TICK      EQU   00H
= 0001                  C  VAL_SET_TICK      EQU   01H
= 0002                  C  VAL_GET_TIME      EQU   02H
= 0003                  C  VAL_SET_TIME      EQU   03H
= 0004                  C  VAL_GET_DATE      EQU   04H
= 0005                  C  VAL_SET_DATE      EQU   05H
= 001A                  C  CLOCKINTERUPT     EQU   1AH
                        C
= 0061                  C  BEEPER_PORT       EQU   0061H               ;speaker port assignment
                        C
= 02D8                  C  PORTCNTL_FDC      EQU   02D8H               ;control port for FDC
                        C
= 0040                  C  PORT0_8254        EQU   0040H               ;counter 0 port adr
= 0041                  C  PORT1_8254        EQU   0041H               ;counter 1 port adr
= 0042                  C  PORT2_8254        EQU   0042H               ;counter 2 port adr
= 0043                  C  PORTCNTL_8254     EQU   0043H               ;counter control port adr
= 0036                  C  CNTR0_INIT        EQU   0036H               ;
= 0068                  C  CNTR1_INIT        EQU   0068H               ;
= 0086                  C  CNTR2_INIT        EQU   0086H               ;
                        C
                        C  ;------This structure shows the format for the date & time as reqd by DOS
                        C  DAYTIME           STRUC
0000  ????               C  DAYS              DW    ?
0002  ??                 C  MINUTES           DB    ?
0003  ??                 C  HOURS             DB    ?
0004  ??                 C  HUNDREDTHS        DB    ?
0005  ??                 C  SECONDS           DB    ?
0006                    C  DAYTIME           ENDS
                        C
                        C  SERIAL_TABLE      STRUC                     ;this structure shows the format of a data
0000  ????               C  PORT_ADDRESS      DW    ?                   ; table for an SCC serial character device.
0002  ????               C  BUFFER_SEG        DW    ?                   ;
0004  ????               C  BUFFER_START      DW    ?                   ;
0006  ????               C  BUFFER_END        DW    ?                   ;
0008  ????               C  BUFFER_HEAD       DW    ?                   ;
000A  ????               C  BUFFER_TAIL       DW    ?                   ;
000C  ??                 C  TIMEOUT           DB    ?                   ;
000D  ??                 C  HANDSHAKE         DB    ?                   ;
000E  ??                 C  WR5COPY           DB    ?                   ;
000F                    C  SERIAL_TABLE      ENDS                      ;
                        C
= 0000                  C  CON_DEV_NUM       EQU   0                   ;these numbers are passed to the bios in
= 0000                  C  SER1_DEV_NUM      EQU   0                   ;   BL to indicate which SCC device is
= 0001                  C  SER2_DEV_NUM      EQU   1                   ;   being accessed.
= 0002                  C  SER3_DEV_NUM      EQU   2                   ;
= 0003                  C  SER4_DEV_NUM      EQU   3                   ;
= 0000                  C  COM1_DEV_NUM      EQU   0
= 0000                  C  LPT1_DEV_NUM      EQU   0
= 0001                  C  LPT2_DEV_NUM      EQU   1
= 0002                  C  LPT3_DEV_NUM      EQU   2
                        C
= 0000                  C  RD_SECTOR         EQU   0                   ;memory disk device driver declarations:
= 0001                  C  WR_SECTOR         EQU   1                   ;read flag
                        C                                              ;write flag
= 000A                  C  DEFAULT_BUFF      EQU   10                  ;system buffers

                        C                INCLUDE IODATA.INC
                        C                SUBTTL Data available to BIOS and IO.SYS
                        C                PAGE 80,132
                        C  ;*************************************************************************
                        C  ;*  OS_CODE segment is the location that IO.SYS will be loaded.          *
                        C  ;*************************************************************************
0000                    C  OS_CODE           SEGMENT AT 0070H
0000                    C  OS_BASE           LABEL   FAR
0000                    C  OS_CODE           ENDS
                        C
```

```
                        ;*****************************************************************************
                        ;*  BOOT_CODE segment is the location that the bootstrap will be loaded.  *
                        ;*****************************************************************************
0000                    BOOT_CODE       SEGMENT AT 1000H
0000                    BOOT_LCN        LABEL   FAR
0000                    BOOT_CODE       ENDS

                        ;*****************************************************************************
                        ;*  Interrupt vector table                                                   *
                        ;*  @@@@ in comment = put in for PC compatibility                            *
                        ;*****************************************************************************
0000                    IODATA          SEGMENT AT 0

                                        ORG     00H                     ;divide by zero vector
0000 ????????           DIVIDE_ERR      DD      ?                       ;
                                                                        ;
                                        ORG     01H * 4                 ;8088 Trap flag
0004 ????????           SINGLE_STEP     DD      ?                       ;
                                                                        ;
                                        ORG     02H * 4                 ;8088 Non-Maskable Interrupt
0008 ????????           NMI             DD      ?                       ;
                                                                        ;
                                        ORG     03H * 4                 ;8088 Single byte interrupt
000C ????????           BREAKPOINT      DD      ?                       ;
                                                                        ;
                                        ORG     04H * 4                 ;8088 Overflow error
0010 ????????           OVERFLOW        DD      ?                       ;
                                                                        ;
                                        ORG     05H * 4                 ;print screen interupt
0014 ????????           PRNT_SCRN       DD      ?                       ;@@@@
                                                                        ;
                                        ORG     08H * 4                 ;time of day interupt
0020 ????????           TICK_TRAP       DD      ?                       ;@@@@

                                        ORG     0CH * 4                 ;trap for COM1 recv char
0030 ????????           COM1IN_TRAP     DD      ?                       ;

                                        ORG     10H * 4                 ;video I/O vector
0040 ????????           VIDEO           DD      ?                       ;@@@@

                                        ORG     11H * 4                                                 ;
0044 ????????           EQUIP_VECT      DD      ?

                                        ORG     12H * 4
0048 ????????           MEMSIZE_VECT    DD      ?

                                        ORG     14H * 4
0050 ????????           COM1_VECT       DD      ?

                                        ORG     16H * 4
0058 ????????           KYBD_VECT       DD      ?

                                        ORG     17H * 4                 ;parallel printer interupt vector
005C ????????           PRINT_VECT      DD      ?

                                        ORG     18H * 4                 ;resident BASIC vector
0060 ????????           BASIC           DD      ?                       ;@@@@
                                                                        ;
                                        ORG     19H * 4
0064 ????????           BOOT_VECT       DD      ?

                                        ORG     1AH * 4                 ;time of day clock trap
0068 ????????           CLOCK_TRAP      DD      ?                       ;

                                        ORG     1BH * 4
006C ????????           KBD_BREAK       DD      ?

                                                                        ;
                                        ORG     1CH * 4                 ;vector to dummy IRET
0070 ????????           TIMER_BREAK     DD      ?                       ;
                                                                        ;
                                        ORG     1DH * 4                 ;video init vector
0074 ????????           VIDEO_INIT      DD      ?                       ;@@@@
                                                                        ;
                                        ORG     1EH * 4                 ;diskette parameters vector
0078 ????????           DISK_PTR        DD      ?                       ;@@@@
                                                                        ;
```

```
007C                    C               ORG     1FH * 4                 ;video graphics vector
007C ????????           C  VIDEO_GRAPH  DD      ?                       ;@@@@
                        C                                               ;
00A4                    C               ORG     29H * 4                 ;MS-DOS INT 29H - write AL to console
00A4 ????????           C  TTY_VECT     DD      ?                       ;
                        C
0108                    C               ORG     42H*4
0108 ????????           C  VIDEO_INT42  DD      ?
                        C                                               ;SCC interrupts:
01C8                    C               ORG     72H*4
01C8 ????????           C  CONINOUT_TRAP DD     ?
                        C
01C4                    C               ORG     71H*4                   ;common SCC BIOS trap
01C4 ????????           C  VECTSCC_TRAP DD      ?                       ;
                        C                                               ;
01CC                    C               ORG     73H*4                   ;timer trap
01CC ????????           C  TIMER_TRAP   DD      ?                       ;
                        C
01DC                    C               ORG     77H * 4                 ;new memory disk interrupt
01DC ????????           C  MDISKINT     DD      ?                       ;
                        C
0208                    C               ORG     82H*4                   ;7314 ch B ext /status hardware interrupt
0208 ????????           C  E14B_TRAP    DD      ?                       ;
                        C                                               ;
020C                    C               ORG     83H*4                   ;7314 ch D ext /status hardware interrupt
020C ????????           C  E14D_TRAP    DD      ?                       ;
                        C                                               ;
0210                    C               ORG     84H*4                   ;7314 ch B rcv hardware interrupt
0210 ????????           C  R14B_TRAP    DD      ?                       ;
                        C                                               ;
0214                    C               ORG     85H*4                   ;7314 ch D rcv hardware interrupt
0214 ????????           C  R14D_TRAP    DD      ?                       ;
                        C                                               ;
0218                    C               ORG     86H*4                   ;7314 ch B spcl rcv hardware interrupt
0218 ????????           C  S14B_TRAP    DD      ?                       ;
                        C                                               ;
021C                    C               ORG     87H*4                   ;7314 ch D spcl rcv hardware interrupt
021C ????????           C  S14D_TRAP    DD      ?                       ;
                        C                                               ;
0228                    C               ORG     8AH*4                   ;7314 ch A ext /status hardware interrupt
0228 ????????           C  E14A_TRAP    DD      ?                       ;
                        C                                               ;
022C                    C               ORG     8BH*4                   ;7314 ch C ext /status hardware interrupt
022C ????????           C  E14C_TRAP    DD      ?                       ;
                        C                                               ;
0230                    C               ORG     8CH*4                   ;7314 ch A rcv hardware interrupt
0230 ????????           C  R14A_TRAP    DD      ?                       ;
                        C                                               ;
0234                    C               ORG     8DH*4                   ;7314 ch C rcv hardware interrupt
0234 ????????           C  R14C_TRAP    DD      ?                       ;
                        C                                               ;
0238                    C               ORG     8EH*4                   ;7314 ch A spcl rcv hardware interrupt
0238 ????????           C  S14A_TRAP    DD      ?                       ;
                        C                                               ;
023C                    C               ORG     8FH*4                   ;7314 ch C spcl rcv hardware interrupt
023C ????????           C  S14C_TRAP    DD      ?                       ;
                        C                                               ;
0240                    C               ORG     90H*4                   ;7863 ch B xmt hardware interupt
0240 ????????           C  X63B_TRAP    DD      ?                       ;
                        C                                               ;
0248                    C               ORG     92H*4                   ;7863 ch B ext /status hardware interrupt
0248 ????????           C  E63B_TRAP    DD      ?                       ;
                        C                                               ;
0250                    C               ORG     94H*4                   ;7863 ch B rcv hardware interrupt
0250 ????????           C  R63B_TRAP    DD      ?                       ;
                        C                                               ;
0258                    C               ORG     96H*4                   ;7863 ch B spcl rcv hardware interrupt
0258 ????????           C  S63B_TRAP    DD      ?                       ;
                        C                                               ;
0268                    C               ORG     9AH*4                   ;7863 ch A ext /status hardware interrupt
0268 ????????           C  E63A_TRAP    DD      ?                       ;
                        C                                               ;
0270                    C               ORG     9CH*4                   ;7863 ch A rcv hardware interrupt
0270 ????????           C  R63A_TRAP    DD      ?                       ;
                        C                                               ;
0278                    C               ORG     9EH*4                   ;7863 ch A spcl rcv hardware interrupt
```

```
0278  ????????              C  S63A_TRAP     DD      ?                       ;
                            C
03FC                        C                ORG     0FFH*4
03FC  ????                  C  MEMTEST       DW      ?
                            C                                                ;
                            C  ;*************************************************************************
                            C  ;*  PC DOS COMPATIBLE ROM BIOS data area                                  *
                            C  ;*************************************************************************
                            C  ;
                            C  ;------RS232 flags
0400                        C                ORG     400H                    ;ROM BIOS data area
0400  0004[                 C  RS232_BASE    DW      4 DUP (?)               ;address of RS232 adapters
          ????              C
                        ]   C
                            C
0408  0004[                 C  PRINTER_BASE  DW      4 DUP (?)               ;address of printers
          ????              C
                        ]   C
                            C
0410  ????                  C  EQUIP_FLAG    DW      ?                       ;installed hardware code
0412  ??                    C  INIT_FLAG     DB      ?                       ;initalization flag
0413  ????                  C  MEM_SIZE      DW      ?                       ;memory size in kbytes
0415  ??                    C  MFG_ERR_FLAG  DB      ?                       ;mailbox for mfg test error codes
0416  ??                    C                DB      ?                       ;
                            C
                            C  ;------KEYBOARD data areas
0417  ??                    C  KBD_FLAG      DB      ?                       ;1st byte of keyboard status
0418  ??                    C  KBD_FLAG_1    DB      ?                       ;2nd byte of keyboard status
0419  ??                    C  ALT_INPUT     DB      ?                       ;storage for alternate kbd input
041A  ????                  C  KBD_BUF_HEAD  DW      ?                       ;ptr to head of kbd bufr
041C  ????                  C  KBD_BUF_TAIL  DW      ?                       ;ptr to tail of kbd bufr
041E  0010[                 C  KBD_BUFFER    DW      16 DUP (?)              ;bufr for up to 15 entries
          ????              C
                        ]   C
                            C
043E                        C  KBD_BUF_END   LABEL   WORD                    ;end of keyboard bufr
                            C
                            C  ;------DISKETTE data areas
043E  ??                    C  SEEK_STATUS   DB      ?                       ;drive recalibration status
043F  ??                    C  MOTOR_STATUS  DB      ?                       ;motor status flag
0440  ??                    C  MOTOR_COUNT   DB      ?                       ;time out counter
0441  ??                    C  DISK_STATUS   DB      ?                       ;return code status byte
0442  0007[                 C  NEC_STATUS    DB      7 DUP (?)               ;status byte from NEC
          ??                C
                        ]   C
                            C
                            C
                            C  ;------VIDEO display data areas
0449  ??                    C  CRT_MODE      DB      ?                       ;current CRT mode
044A  ????                  C  CRT_COLS      DW      ?                       ;number of columns on screen
044C  ????                  C  CRT_LENGTH    DW      ?                       ;length of regen in bytes
044E  ????                  C  CRT_START     DW      ?                       ;start addr in regen bufr
0450  0008[                 C  CURS_POSITION DW      8 DUP (?)               ;cursor pos for up to 8 pages
          ????              C
                        ]   C
                            C
0460  ????                  C  CURS_MODE     DW      ?                       ;current curs mode code
0462  ??                    C  ACTIVE_PAGE   DB      ?                       ;current page on screen
0463  ????                  C  ADDR_6845     DW      ?                       ;base addr for disp card
0465  ??                    C  CRT_MODE_SET  DB      ?                       ;cur code of 3 x 8 register
0466  ??                    C  CRT_PALETTE   DB      ?                       ;current crt color card
                            C
                            C  ;------POST data areas
0467  ????                  C  IO_ROM_INIT   DW      ?                       ;ptr to optional i/o rom init routine
0469  ????                  C  IO_ROM_SEG    DW      ?                       ;ptr to rom segment
046B  ??                    C  INTER_FLAG    DB      ?                       ;interupt occured flag
                            C
                            C  ;------TIMER data areas
046C  ????                  C  TIMER_LOW     DW      ?                       ;low word of timer count
046E  ????                  C  TIMER_HIGH    DW      ?                       ;high word of timer count
0470  ??                    C  TIMER_OFLW    DB      ?                       ;timer rollover flag
                            C
                            C  ;------system DATA AREAS
0471  ??                    C  BIOS_BREAK    DB      ?                       ;bit 7 = 1 if break key hit
0472  ????                  C  RESET_FLAG    DW      ?                       ;word = 1234 if kbd reset underway
                            C
                            C  ;------FIXED DISK data areas
```

```
0474  ????                C  SCSI_INT_ADR   DW      ?                       ;
0476  ????                C  SCSI_INT_SEG   DW      ?                       ;
                          C
                          C  ;------PRINT & RS232 timeout variables
0478  0004[               C  PRNT_TIMEOUT   DB      4 DUP (?)               ;
        ??                C
                        ]   C
                          C
047C  0004[               C  RS232_TIMEOUT DB       4 DUP (?)               ;
        ??                C
                        ]   C
                          C
                          C
                          C  ;------more KEYBOARD data areas
0480  ????                C  BUFF_START     DW      ?                       ;
0482  ????                C  BUFF_END       DW      ?                       ;
                          C
0484  0007[               C                 DB      7 DUP (?)               ;unused ram
        ??                C
                        ]   C
                          C
                          C
                          C  ;------more FLOPPY data areas
048B  ????                C  RES_BIOS_FLAG DW       ?
                          C
                          C  ;------more HARD DISK data areas
048D  ????                C  DISK_DATA_SEG DW       ?
048F  ??                  C  PHYS_FMT_FLAG DB       ?
0490  ??                  C  HF_CNTL        DB      ?
                          C
                          C  ;------more DISKETTE data areas
0491  ??                  C  RECAL_MAP      DB      ?
0492  ??                  C  DRV_TYPE_TBL   DB      ?
0493  ??                  C                 DB      ?
0494  ??                  C                 DB      ?
0495  ??                  C  DISK_TRK       DB      ?
0496  ??                  C                 DB      ?
0497  ??                  C                 DB      ?
                          C
                          C  ;------KEYBOARD led flag
0498  ??                  C  KBD_FLAG_2     DB      ?
                          C
                          C  ;------RTC data areas
0499  ????                C  RTC_USER_ADDR DW       ?
049B  ????                C  RTC_USER_SEG  DW       ?
049D  ????                C  RTC_LOWCOUNT  DW       ?
049F  ????                C  RTC_HICOUNT   DW       ?
04A1  ??                  C  RTC_WAIT      DB       ?
                          C
                          C  ;*************************************************************************
                          C  ;*  STD DOS data areas                                                   *
                          C  ;*************************************************************************
0500                      C                 ORG     500H
0500  ??                  C  STATUS_BYTE    DB      ?
0501  000F[               C                 DB      15 DUP (?)              ;reserved area
        ??                C
                        ]   C
                          C
                          C
0510  ????????            C  PTRSAVE        DD      ?                       ;request block pointer
                          C
0514  ????                C  DAYSCOUNTED    DW      ?                       ;for use with clock routines
                          C
                          C  ;do not separate the following six declarations!
0516  000F[               C  SER1_DATA      DB      (SIZE SERIAL_TABLE) DUP(?) ;
        ??                C
                        ]   C
                          C
0525  000F[               C  SER2_DATA      DB      (SIZE SERIAL_TABLE) DUP(?) ;
        ??                C
                        ]   C
                          C
0534  000F[               C  SER3_DATA      DB      (SIZE SERIAL_TABLE) DUP(?) ;
        ??                C
                        ]   C
                          C
```

```
0543  000F[                C  SER4_DATA     DB      (SIZE SERIAL_TABLE) DUP(?) ;
        ??                  C
                          ]   C
                              C
                              C
0552  0020[                 C  BUF1          DB      32 DUP(?)                  ;32-byte input buffers for
        ??                  C
                          ]   C
                              C
0572                          C  BUF1_END      LABEL   BYTE                       ;     SCC character devices
0572  0020[                 C  BUF2          DB      32 DUP(?)                  ;
        ??                  C
                          ]   C
                              C
0592                          C  BUF2_END      LABEL   BYTE                       ;
0592  0020[                 C  BUF3          DB      32 DUP(?)                  ;
        ??                  C
                          ]   C
                              C
05B2                          C  BUF3_END      LABEL   BYTE                       ;
05B2  0020[                 C  BUF4          DB      32 DUP(?)                  ;
        ??                  C
                          ]   C
                              C
05D2                          C  BUF4_END      LABEL   BYTE                       ;
                              C
05D2  ??                      C  COM1CHAR      DB      ?
05D3  ??                      C  COM1READY     DB      ?
                              C
05D4  ??                      C  INNULSEQ      DB      ?
05D5  ??                      C  BRKFLAG       DB      ?
05D6  ??                      C  SCANCODE      DB      ?
                              C
05D7  ??                      C  SCCFLAG       DB      ?
05D8  ??                      C  CON_RESET_ENABLE      DB ?                       ;enables ESC c reset from CON
                              C
05D9  0040[                   C                DW      64 DUP(?)                  ;temporary stack area for device drivers
        ????                  C
                          ]   C
                              C
0659                          C  DRVSTACK      LABEL   WORD                       ;
0659  0020[                   C                DW      32 DUP(?)                  ;temporary stack area for hdw interrupts
        ????                  C
                          ]   C
                              C
0699                          C  INTSTACK      LABEL   WORD                       ;
                              C
0699  ????                    C  STACK_SP      DW      ?                          ;temporary SP storage for device drivers
069B  ????                    C  STACK_SS      DW      ?                          ;temporary SS storage for device drivers
069D  ????                    C  INTR_SP       DW      ?                          ;temporary SP storage for hdw interrupts
069F  ????                    C  INTR_SS       DW      ?                          ;temporary SS storage for hdw interrupts
                              C
06A1  ??                      C  DRVCOUNT      DB      ?                          ;This mailbox is for use by
                              C                                                   ;DISKDRVR to determine how
                              C                                                   ;many drives are installed
                              C                                                   ;before it. MSDOS didn't stuff
                              C                                                   ;the DRIVENUMBER mailbox in
                              C                                                   ;the request header.
                              C
06A2  ??                      C  RAMDRVBFLG    DB      ?
06A3  ????                    C  MEMDSKMASK    DW      ?
06A5  ??                      C  ESC_FLAG      DB      ?
06A6  ????                    C  CLS_PTR       DW      ?
                              C
06A8                          C  IODATAEND     LABEL   BYTE                       ;
                              C
06A8                          C  IODATA        ENDS

                                 SUBTTL  BIOS Equates and Definitions
= 0020                           ROMSIZE     EQU     32                         ;BIOS ROM size in Kbytes
= 0039                           PORT7712    EQU     039H                       ;7712 parity reset port

0000                             DIAGNOSTICS SEGMENT AT 0D800H                  ;location of diagnostics EPROM
= 5A5A                           DIAG_ID     EQU     5A5AH                      ;diagnostics ID word
0000                                         ORG     0                          ;
0000  ????                       DIAG_ID_LOC DW      ?                          ;diagnostics ID word location
```

```
0002                    DIAG            PROC    FAR                     ;start of diagnostics
                        DIAG            ENDP
0002                    DIAGNOSTICS     ENDS                            ;
                                                                        ;

                        SUBTTL  Main BIOS code

0000                    CODE            SEGMENT PUBLIC
                                        ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
                        ;*************************************************************************
                        ;* BIOS_START                                                            *
                        ;*              This is the start of the BIOS ROM.  A jump to BIOS_START is *
                        ;*              executed at reset.                                       *
                        ;*                                                                       *
                        ;* Entry:       none                                                     *
                        ;* Exit:        AX = base segment of MSDOS.SYS                           *
                        ;*              BX = segment of last RAM address                         *
                        ;*              CL = default drive                                       *
                        ;* Alters:      ALL                                                      *
                        ;*************************************************************************
0000                    BIOS_START      PROC    FAR                     ;
0000  EB 3A 90                         JMP     BIOS                    ;
0003  53 54 44 2D 44 4F 53              DB      'STD-DOS VERSION 2.00'
      20 56 45 52 53 49 4F
      4E 20 32 2E 30 30
0017  28 43 29 43 6F 70 79              DB      '(C)Copyright 1986, 1987 Pro-Log Corp.'
      72 69 67 68 74 20 31
      39 38 36 2C 20 31 39
      38 37 20 50 72 6F 2D
      4C 6F 67 20 43 6F 72
      70 2E
003C  FA                        BIOS:   CLI                             ;disable interrupts during init
003D  8B ---- R                         MOV     AX,SEG DIAGNOSTICS      ;
0040  8E D8                             MOV     DS,AX                   ;DS = segment of diagnostics ROM
                                        ASSUME  DS:DIAGNOSTICS          ;
0042  81 3E 0000 R 5A5A                CMP     DIAG_ID_LOC,DIAG_ID     ;check for ID word
0048  75 05                             JNE     BIOS2                   ;jump if no ID found
004A  EA 0002 ---- R                    JMP     DIAG                    ;call the diagnostics EPROM
004F  FA                        BIOS2:  CLI                             ;turn ints off in case diag turned on
0050  FC                                CLD                             ;always increment string operations
0051  33 C0                             XOR     AX,AX                   ;
0053  8B F8                             MOV     DI,AX                   ;
0055  8E C0                             MOV     ES,AX                   ;
                                        ASSUME  ES:NOTHING              ;ES:DI = interrupt vector area
0057  8D 1E 17C5 R                     LEA     BX,INTR_ERR             ;
005B  8C CA                             MOV     DX,CS                   ;DX:BX = INTR_ERR address
005D  B9 0100                           MOV     CX,0100H                ;0100H interrupt vectors to initialize
0060  8B C3                     B0:     MOV     AX,BX                   ;get offset
0062  AB                                STOSW                           ;store INTR_ERR offset
0063  8B C2                             MOV     AX,DX                   ;get segment
0065  AB                                STOSW                           ;store INTR_ERR segment
0066  E2 F8                             LOOP    B0                      ;repeat for all interrupt vectors
0068  B8 ---- R                         MOV     AX,SEG IODATA           ;
006B  8E D8                             MOV     DS,AX                   ;
                                        ASSUME  DS:IODATA               ;
006D  C7 06 0008 R 17C4 R               MOV     WORD PTR NMI,OFFSET IRET_INST ;NMI goes to an IRET
0073  8C 0E 000A R                      MOV     WORD PTR NMI+2,CS       ;
0077  A1 0413 R                         MOV     AX,MEM_SIZE             ;get & save prev mem size for
007A  A3 03FC R                         MOV     MEMTEST,AX              ;test for warmup delay
007D  E6 39                             OUT     PORT7712,AL             ;output to 7712 port to clear parity error
007F  B8 0000                           MOV     AX,0                    ;
0082  B9 7E00                           MOV     CX,8000H-0200H          ;clear remaining 64K
0085  F3/ AB                            REP     STOSW                   ;clear lowest 64K of RAM
0087  B8 ---- R                         MOV     AX,SEG IODATA           ;set SS = iodata segment
008A  8E D0                             MOV     SS,AX                   ;
008C  BC 0659 R                         MOV     SP,OFFSET DRVSTACK      ;BIOS uses device driver stack
008F  E8 0410 R                         CALL    FIND_RAM_TOP            ;initializes BP (memory size)
0092  E8 062C R                         CALL    INIT_MEMSIZE            ;init memsize flag
0095  E8 0452 R                         CALL    CLEAR_RAM               ;clear RAM up to BP (memory size)
0098  B0 A5                             MOV     AL,0A5H ;set clock hi on kybd if present
009A  E6 61                             OUT     KYBD_PORT,AL            ;
009C  E6 39                             OUT     PORT7712,AL             ;output to 7712 port to clear parity error
009E  C7 06 0008 R 17AB R               MOV     WORD PTR NMI,OFFSET PARITY_ERROR ;initialize NMI
00A4  C7 06 01DC R 0334 R               MOV     WORD PTR MDISKINT,OFFSET MIO;initialize memory disk
00AA  8C 0E 01DE R                      MOV     WORD PTR MDISKINT+2,CS  ;  interrupt
00AE  B8 0004                           MOV     AX,0004H                ;init equip flag = 0004
00B1  A3 0410 R                         MOV     EQUIP_FLAG,AX           ;(sys ram = 128k)
```

```
00B4  E8 0657 R                 CALL    INIT_HDWR               ;init 7888 cpu chips & vectors
00B7  E8 0208 R                 CALL    CHK_VIDEO               ;find & call video bios extensions
00BA  E8 04DA R                 CALL    SCCINIT                 ;initialize the SCCs
00BD  E8 0589 R                 CALL    CHKCONSOLE              ;redirect console if required
00C0  8D 36 194F R              LEA     SI,BIOSMSG              ;
00C4  E8 0487 R                 CALL    MESSAGE                 ;send BIOS message
00C7  E8 0220 R                 CALL    CHK_BIOS                ;find & call any more extended bios
00CA  A1 03FC R                 MOV     AX,MEMTEST              ;
00CD  C7 06 03FC R 17C5 R       MOV     MEMTEST,OFFSET INTR_ERR ;
00D3  39 06 0413 R              CMP     MEM_SIZE,AX             ;if mem size compares, skip POST
00D7  74 0D                     JE      SKIP_POST               ;
00D9  E4 FF                     IN      AL,FFPORT               ;
00DB  3C AA                     CMP     AL,0AAH                 ;if port in FF = AA, skip test
00DD  74 07                     JE      SKIP_POST               ;
00DF  A1 048B R                 MOV     AX,RES_BIOS_FLAG        ;
00E2  24 04                     AND     AL,SCSIFLAG             ;
00E4  75 1D                     JNZ     DOPOST                  ;if no scsi installed, skip POST
00E6  8D 36 18CD R      SKIP_POST:      LEA     SI,CRLF         ;do a CR/LF
00EA  E8 0487 R                 CALL    MESSAGE                 ;
00ED  E8 019E R                 CALL    SHOWMEMSIZE             ;show amount of sys ram tested
00F0  8D 36 18CD R              LEA     SI,CRLF                 ;
00F4  E8 0487 R                 CALL    MESSAGE                 ;
00F7  E8 0285 R         DOBOOT:         CALL    LOADBOOTSECT    ;load a valid boot sector
00FA  8D 36 1863 R      FATAL_ERROR:    LEA     SI,SYSERROR     ;if return, error
00FE  E8 0487 R                 CALL    MESSAGE                 ;send error message
0101  EB FE             ERROR_STOP:     JMP     ERROR_STOP      ;INFINITE LOOP!
                        BIOS_START      ENDP                    ;

0103  E8 014D R         DOPOST:         CALL    POST            ;do power on self test
0106  E8 02E5 R                 CALL    CHK_ESC                 ;set flag if ESC received
0109  E8 019E R                 CALL    SHOWMEMSIZE             ;
010C  E8 01AD R                 CALL    ISTESTEDMSG             ;
010F  A1 0413 R                 MOV     AX,MEM_SIZE             ;post takes approx 1 sec for 32k
0112  B1 20                     MOV     CL,32                   ;so set DX = 15 - (#K / 32)
0114  F6 F1                     DIV     CL                      ;
0116  B4 0F                     MOV     AH,15                   ;
0118  2A E0                     SUB     AH,AL                   ;
011A  BA 0000                   MOV     DX,0                    ;
011D  72 D8                     JC      DOBOOT                  ;if delay reqd < 0, go to boot
011F  8A D4                     MOV     DL,AH                   ;
0121  E8 0126 R                 CALL    SCSIDELAY               ;do delay for scsi
0124  EB D1                     JMP     SHORT DOBOOT            ;skip CR/LF
                        ;------Delay about one sec for every count in DX
0126                    SCSIDELAY       PROC    NEAR            ;
0126  F7 C2 0001                TEST    DX,1                    ;make DX an even number
012A  74 01                     JZ      INIT_DELAY              ;
012C  42                        INC     DX                      ;
012D  E8 013B R         INIT_DELAY:     CALL    DELAYMESG       ;show or clear delay msg
0130  B9 0000                   MOV     CX,0                    ;
0133  50                DELAY_LOOP:     PUSH    AX              ;
0134  58                               POP     AX              ;
0135  E2 FC                            LOOP    DELAY_LOOP      ;
0137  4A                        DEC     DX                      ;
0138  75 F3                     JNZ     INIT_DELAY              ;
013A  C3                        RET                             ;
                        SCSIDELAY       ENDP                    ;

                        ;------DELAYMESG will show delay mesg if DX is even, or clr mesg if odd.
013B                    DELAYMESG       PROC    NEAR            ;
013B  8D 36 19E6 R              LEA     SI,DELAYMSG             ;set ptr = delay msg
013F  F7 C2 0001                TEST    DX,1                    ;
0143  74 04                     JZ      DELMSGOUT               ;if DX is even, show msg
0145  8D 36 1A06 R              LEA     SI,CLEARMSG             ;set ptr = spaces
0149  E8 0487 R         DELMSGOUT:     CALL    MESSAGE         ;
014C  C3                        RET                             ;
                        DELAYMESG       ENDP                    ;

                        ;*****************************************************************************
                        ;*                                                                           *
                        ;* POST                                                                      *
                        ;*              The power on self test will perform a system memory check    *
                        ;*              and display the size of memory sucessfully tested in kbytes.*
                        ;*              The best-guess equipment list will be displayed also.        *
                        ;* ENTRY:       MEM_SIZE mailbox set by "FIND_RAM_TOP"                       *
                        ;* ALTERS:      all except seg regs                                          *
                        ;*****************************************************************************
                                ASSUME  CS:CODE,DS:IODATA,ES:NOTHING,SS:NOTHING
```

```
 014D                   POST    PROC    NEAR                    ;
 014D  E8 0196 R                CALL    MEMTESTMSG              ;show mem testing message
 0150  BB 0000                  MOV     BX,0                    ;init # k tested = 0
 0153  BA 0000                  MOV     DX,0                    ;init test segment = 0
 0156  E8 0185 R        MEMTESTLOOP:    CALL    SHOWKBYTES      ;show # kbytes tested so far
 0159  E8 016D R                CALL    TESTBLOCK               ;test a 1k block
                                ASSUME  DS:IODATA               ;
 015C  72 0A                    JC      MEMSIZE                 ;if error, exit
 015E  83 C2 40                 ADD     DX,64                   ;add 1k to test segment
 0161  43                       INC     BX                      ;incr # k tested
 0162  3B 1E 0413 R             CMP     BX,MEM_SIZE             ;
 0166  72 EE                    JC      MEMTESTLOOP             ;if not to ram top, loop
 0168  89 1E 0413 R     MEMSIZE:        MOV     MEM_SIZE,BX     ;store mem size if not ok
 016C  C3                       RET                             ;
                        POST    ENDP                            ;

                        ;------TESTBLOCK will perform a non-destructive memory test of a 1k block
                        ;       starting at segment DX. CF set on return = error in test.
 0160                   TESTBLOCK       PROC    NEAR
                                ASSUME  CS:CODE,DS:IODATA,ES:NOTHING,SS:NOTHING
 016D  1E                       PUSH    DS
 016E  B9 0400                  MOV     CX,1024                 ;set count = 1k
 0171  8E DA                    MOV     DS,DX                   ;set DS = test segment
                                ASSUME  DS:NOTHING              ;
 0173  BE 0000                  MOV     SI,0                    ;init pointer = 0
 0176  FB               TESTBLKLOOP:    STI                     ;
 0177  90                       NOP                             ;interupt window
 0178  FA                       CLI                             ;
 0179  8A 24                    MOV     AH,[SI]                 ;get byte @SI & save in AH
 017B  C6 04 AA                 MOV     BYTE PTR[SI],0AAH       ;test if AA can be written
 017E  80 3C AA                 CMP     BYTE PTR[SI],0AAH       ;
 0181  F9                       STC                             ;
 0182  75 0F                    JNE     TESTBLKXIT              ;exit if error
 0184  C6 04 55                 MOV     BYTE PTR[SI],055H       ;test if 55 can be written
 0187  80 3C 55                 CMP     BYTE PTR[SI],055H       ;
 018A  F9                       STC                             ;
 018B  75 06                    JNE     TESTBLKXIT              ;exit if error
 018D  88 24                    MOV     [SI],AH                 ;restore byte
 018F  46                       INC     SI                      ;incr pointer
 0190  E2 E4                    LOOP    TESTBLKLOOP             ;loop till 1k done
 0192  F8                       CLC                             ;return NC = tested good
 0193  1F               TESTBLKXIT:     POP     DS              ;
 0194  FB                       STI                             ;interrupts back on
 0195  C3                       RET                             ;
                        TESTBLOCK       ENDP                    ;

                        ;------MEMTESTMSG will show mem test in progress message
                                ASSUME  CS:CODE,DS:IODATA,ES:NOTHING,SS:NOTHING
 0196                   MEMTESTMSG      PROC    NEAR                    ;show mem test in progress message
 0196  8D 36 1995 R             LEA     SI,MEMTESTMESG          ;
 019A  E8 0487 R                CALL    MESSAGE                 ;
 019D  C3                       RET                             ;
                        MEMTESTMSG      ENDP                    ;

                        ;------SHOWMEMSIZE will convert #k sucessfully tested and show it
                                ASSUME  CS:CODE,DS:IODATA,ES:NOTHING,SS:NOTHING
 019E                   SHOWMEMSIZE     PROC    NEAR            ;
 019E  8B 1E 0413 R             MOV     BX,MEM_SIZE             ;get MEM_SIZE to BX
 01A2  E8 0185 R                CALL    SHOWKBYTES              ;go convert & show BX
 01A5  8D 36 198B R             LEA     SI,TESTEDMSG            ;
 01A9  E8 0487 R                CALL    MESSAGE                 ;
 01AC  C3                       RET                             ;
                        SHOWMEMSIZE     ENDP                    ;

                        ;------ISTESTEDMSG will simply show "Have Been Tested"
 01AD                   ISTESTEDMSG     PROC    NEAR            ;
 01AD  8D 36 19D2 R             LEA     SI,ISTESTEDMESG         ;
 01B1  E8 0487 R                CALL    MESSAGE                 ;
 01B4  C3                       RET                             ;
                        ISTESTEDMSG     ENDP                    ;

                        ;------SHOWKBYTES will show contents of BX to screen
 01B5                   SHOWKBYTES      PROC    NEAR            ;
 01B5  B0 0D                    MOV     AL,CR                   ;
 01B7  CD 29                    INT     29H                     ;move curs to start of cur line
 01B9  53                       PUSH    BX                      ;
```

```
01BA  E8 01C9 R                 CALL    HEXTOBCD                ;convert BX to BCD to AX
01BD  86 C4                     XCHG    AL,AH                   ;
01BF  E8 04AF R                 CALL    HEX_OUT_NIBL            ;convert to ascii & show
01C2  86 C4                     XCHG    AL,AH                   ;
01C4  E8 0498 R                 CALL    HEX_OUT_BYTE            ;
01C7  5B                        POP     BX                      ;
01C8  C3                        RET                             ;
                        SHOWKBYTES      ENDP                    ;

                        ;*************************************************************************
                        ;* HEXTOBCD                                                              *
                        ;*           This routine will convert the hex values in BX to BCD       *
                        ;* ENTRY:    BX = hex value                                              *
                        ;* EXIT:     AX = BCD value                                              *
                        ;*************************************************************************
01C9                    HEXTOBCD        PROC    NEAR                    ;
                                        ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
01C9  51                               PUSH    CX                      ;
01CA  53                               PUSH    BX                      ;
01CB  8B CB                            MOV     CX,BX                   ;set CX = hex value
01CD  BB 0000                          MOV     BX,0                    ;init BCD to 0000
01D0  83 F9 00          HEXTOBCDLOOP:  CMP     CX,0                    ;
01D3  74 2E                            JE      XITHEXTOBCD             ;if CX = 0, exit
01D5  43                               INC     BX                      ;inc BCD value
01D6  8A C3                            MOV     AL,BL                   ;
01D8  24 0F                            AND     AL,0FH                  ;
01DA  3C 0A                            CMP     AL,0AH                  ;
01DC  74 03                            JE      ONESOF                  ;if ones overflow...
01DE  49                CONTHEXTOBCD:  DEC     CX                      ;decr hex count & loop
01DF  EB EF                            JMP     SHORT HEXTOBCDLOOP      ;
01E1  80 E3 F0          ONESOF:        AND     BL,0F0H                 ;clear ones diget
01E4  80 C3 10                         ADD     BL,10H                  ;add 1 to tens diget
01E7  8A C3                            MOV     AL,BL                   ;
01E9  24 F0                            AND     AL,0F0H                 ;
01EB  3C A0                            CMP     AL,0A0H                 ;
01ED  75 EF                            JNE     CONTHEXTOBCD            ;if tens not overflow, continue
01EF  B3 00                            MOV     BL,0                    ;set ten diget to 0
01F1  FE C7                            INC     BH                      ;incr hundreds diget
01F3  8A C7                            MOV     AL,BH                   ;
01F5  24 0F                            AND     AL,0FH                  ;
01F7  3C 0A                            CMP     AL,0AH                  ;
01F9  75 E3                            JNE     CONTHEXTOBCD            ;if hundreds not overflow, continue
01FB  80 E7 F0                         AND     BH,0F0H                 ;clr hundreds diget
01FE  80 C7 10                         ADD     BH,10H                  ;add 1 to thousands diget
0201  EB DB                            JMP     SHORT CONTHEXTOBCD      ;
0203  8B C3             XITHEXTOBCD:   MOV     AX,BX                   ;move BCD to AX
0205  5B                               POP     BX                      ;
0206  59                               POP     CX                      ;
0207  C3                               RET                             ;
                        HEXTOBCD        ENDP                            ;

                        ;*************************************************************************
                        ;* CHK_VIDEO                                                             *
                        ;*           This routine will search for a video bios extension         *
                        ;*           starting at "VIDEOSTART" and finishing at "VIDEOEND".       *
                        ;*           If a valid extension is found, the INSTALL function         *
                        ;*           will be executed.                                           *
                        ;*************************************************************************
                                        ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
0208                    CHK_VIDEO       PROC    NEAR                    ;
0208  BE C000                          MOV     SI,VIDEOSTART           ;set SI,DI = start & end segment
020B  BF C800                          MOV     DI,VIDEOEND             ;
020E  E8 022A R                        CALL    EXTND_BIOS              ;find & execute bios extension
0211  1E                               PUSH    DS                      ;
0212  E8 03F1 R                        CALL    DDS                     ;
0215  BF 0108 R                        MOV     DI,OFFSET VIDEO_INT42   ;set video int 42h vector to IRET
0218  BB 06FA R                        MOV     BX,OFFSET DUMMY_IRET    ;
021B  E8 12C3 R                        CALL    MOVEADDR                ;
021E  1F                               POP     DS                      ;MS byte of seg for int 09)
021F  C3                               RET                             ;
                        CHK_VIDEO       ENDP                            ;

                        ;*************************************************************************
                        ;* CHK_BIOS                                                              *
                        ;*           This routine will search for a valid bios extension         *
                        ;*           starting at "BIOSTART" and finishing at "BIOSEND".          *
```

```
                        ;*          If a valid extension is found, the INSTALL function        *
                        ;*          will be executed.                                          *
                        ;*******************************************************************
0220                            ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
0220  BE C800           CHK_BIOS    PROC    NEAR                    ;
0223  BF E000                      MOV     SI,BIOSTART             ;set SI,DI = start & end segment
0226  E8 022A R                    MOV     DI,BIOSEND              ;
0229  C3                          CALL    EXTND_BIOS              ;find & execute bios extension
                                  RET                             ;
                        CHK_BIOS    ENDP                          ;

                        ;*******************************************************************
                        ;* EXTND_BIOS                                                      *
                        ;*          This routine will start searching at segment (SI) offset 0  *
                        ;*          to find a legal bios extension code (AA55). If one is found,*
                        ;*          the INSTALL routine (@ fnd adr + 3) will be called.        *
                        ;* ENTRY:   SI=START,DI=END SEG                                     *
                        ;* EXIT:    none                                                    *
                        ;* ALTERS:  all                                                     *
                        ;* CALLS:   INSTALL                                                 *
                        ;*******************************************************************
022A                            ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
022A  8B C6             EXTND_BIOS  PROC    NEAR                ;
022C  8E C0                         MOV     AX,SI               ;set ES = 1st bios prom seg
022E  33 DB                         MOV     ES,AX               ;
0230  26: 81 3F AA55    CHK_EXTND:  XOR     BX,BX               ;set BX = offset 0
0235  B9 0080                       CMP     ES:[BX],EXTND_BIOS_ID ;see if bios prom ID valid
0238  75 21                         MOV     CX,128              ;set CL = 2k
023A  83 C3 03                      JNE     NXT_EXT_BIO         ;if not, continue check
023D  E8 0266 R                     ADD     BX,3                ;
0240  33 DB                         CALL    RUTNE_ESBX          ;go call bios ext routine
0242  83 C3 02                      XOR     BX,BX               ;
0245  26: 8A 07                     ADD     BX,2                ;
0248  32 E4                         MOV     AL,BYTE PTR ES:[BX] ;get # sectors in bios ext
024A  B3 04                         XOR     AH,AH               ;
024C  F6 F3                         MOV     BL,4                ;
024E  80 FC 00                      DIV     BL                  ;adj to 2k increment
0251  74 02                         CMP     AH,0                ;
0253  FE C0                         JE      CONTINUE            ;
0255  B4 80                         INC     AL                  ;
0257  F6 E4             CONTINUE:   MOV     AH,128              ;CX = # paragraphs
0259  8B C8                         MUL     AH                  ;
025B  8C C0                         MOV     CX,AX               ;
025D  03 C1             NXT_EXT_BIO: MOV    AX,ES               ;
025F  8E C0                         ADD     AX,CX               ;add # paragraphs to ES
0261  38 C7                         MOV     ES,AX               ;
0263  75 C9                         CMP     AX,DI               ;
0265  C3                            JNE     CHK_EXTND           ;if end of bios prom area, exit
                                    RET                         ;
                        EXTND_BIOS  ENDP                        ;

                        ;------Call routine @ES:BX in bios extension
0266                    RUTNE_ESBX  PROC    NEAR                ;
0266  06                           PUSH    ES                  ;save registers
0267  1E                           PUSH    DS                  ;
0268  57                           PUSH    DI                  ;
0269  55                           PUSH    BP                  ;
026A  B8 1000                       MOV     AX,1000H            ;temp ram @ 1000:0000
026D  8E D8                         MOV     DS,AX               ;
026F  BE 0000                       MOV     SI,0                ;
0272  89 1C                         MOV     [SI],BX             ;store offset (BX)
0274  83 C6 02                      ADD     SI,2                ;
0277  8C C0                         MOV     AX,ES               ;
0279  89 04                         MOV     [SI],AX             ;store segment (ES)
027B  83 EE 02                      SUB     SI,2                ;
027E  FF 1C                         CALL    DWORD PTR[SI]       ;make CALL to bios routine
0280  5D                            POP     BP                  ;restore registers
0281  5F                            POP     DI                  ;
0282  1F                            POP     DS                  ;
0283  07                           POP     ES                  ;
0284  C3                           RET                         ;continue
                        RUTNE_ESBX  ENDP

                        ;*******************************************************************
                        ; LOADBOOTSECT                                                     *
                        ;           First, will allow floppy to boot system               *
```

```
                        ;               If no floppy boot, will allow SCSI to boot system          *
                        ;               if no SCSI boot, will try to boot from mem disk            *
                        ;               If a valid boot is found, this routine will call it and    *
                        ;               never return.                                              *
                        ;                                                                          *
                        ; Entry:      BP = mem size (?)                                            *
                        ; Exit:       C0 = valid boot sector loaded                                *
                        ;               C1 = failure                                               *
                        ; Alters:                                                                  *
                        ;*****************************************************************************
0285                    LOADBOOTSECT  PROC    NEAR                    ;
                                      ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
0285  E8 03F1 R                      CALL    DDS                     ;set DS = IODATA
                                      ASSUME  DS:IODATA               ;
0288  A1 048B R                      MOV     AX,RES_BIOS_FLAG        ;get & chk flag for floppy
028B  24 10                          AND     AL,FLOPPYFLAG           ;
028D  75 10                          JNZ     FLOPPYBOOT              ;if floppy installed, boot from it
028F  80 3E 06A5 R 01  FLOPPYRETURN: CMP     ESC_FLAG,01H            ;chk if flag set
0294  74 21                          JE      MEMDSKBOOT              ;if so, dont try SCSI boot
0296  A1 048B R                      MOV     AX,RES_BIOS_FLAG        ;get & chk flag for scsi
0299  24 04                          AND     AL,SCSIFLAG             ;
029B  75 0F                          JNZ     SCSIBOOT                ;if scsi installed, boot from it
029D  EB 18                          JMP     SHORT MEMDSKBOOT        ;go try to boot from mem disk

029F  BB 0006          FLOPPYBOOT:   MOV     BX,BOOTOFFSET           ;set BX = offset (boot location)
02A2  B8 CC00                        MOV     AX,FLOPPYSEG            ;set ES = floppy bios extension seg
02A5  8E C0                          MOV     ES,AX                   ;
02A7  E8 0266 R                      CALL    RUTNE_ESBX              ;call routine at ES:BX
02AA  EB E3                          JMP     SHORT FLOPPYRETURN      ;if it returns, it didn't boot

02AC  BB 0006          SCSIBOOT:     MOV     BX,BOOTOFFSET           ;set BX = offset (boot location)
02AF  B8 C800                        MOV     AX,SCSISEG              ;set ES = scsi bios extension seg
02B2  8E C0                          MOV     ES,AX                   ;
02B4  E8 0266 R                      CALL    RUTNE_ESBX              ;call routine at ES:BX
                                                                     ;if it returns, it didn't boot
                            ;----- Try to boot from mem disk
02B7  0E               MEMDSKBOOT:   PUSH    CS                      ;set DS = CODE
02B8  1F                             POP     DS                      ;
                                     ASSUME  DS:CODE                 ;
02B9  B2 00                          MOV     DL,0                    ;1st memory disk
02BB  B8 ---- R                      MOV     AX,SEG BOOT_CODE        ;set xfer seg
02BE  8E C0                          MOV     ES,AX                   ;
                                     ASSUME  ES:BOOT_CODE            ;

                            ;----- attempt to read a valid boot sector:
02C0  B9 0000          LB1:          MOV     CX,0                    ;one sector
02C3  BB 0000                        MOV     BX,0                    ;set xfer offset
02C6  B8 0201                        MOV     AX,0201H                ;read one sector
02C9  CD 77                          INT     MDISK_TRAP              ;
02CB  72 07                          JC      LB2                     ;failure - next drive
02CD  8C C1                          MOV     CX,ES                   ;
02CF  E8 0432 R                      CALL    FIND_DISK               ;check for P-L ID string
02D2  75 09                          JNZ     LB_SUCCESS              ;jump if valid boot sector
02D4  FE C2            LB2:          INC     DL                      ;next drive
02D6  80 FA 10                       CMP     DL,16                   ;checked all drives?
02D9  72 E5                          JB      LB1                     ;   -jump if not
02DB  EB 07                          JMP     SHORT LB_FAIL           ;checked 'em all - FAILED
02DD  9A 0000 ---- R   LB_SUCCESS:   CALL    BOOT_LCN                ;C0 = bootstrap loaded
02E2  EB F0                          JMP     SHORT LB2               ;
02E4  C3               LB_FAIL:      RET                             ;
                       LOADBOOTSECT  ENDP                            ;

                       ; This routine will set ESC_FLAG if an ESC was received during POST
                                     ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
02E5                   CHK_ESC       PROC    NEAR                    ;
02E5  E8 03F1 R                      CALL    DDS                     ;make sure DS = iodata
02E8  1E                             PUSH    DS                      ;
02E9  8B 1E 041A R                   MOV     BX,KBD_BUF_HEAD         ;set BX = kbd buffer head
02ED  8B 0E 041C R                   MOV     CX,KBD_BUF_TAIL         ;set CX = kbd bufr tail
02F1  A1 048B R                      MOV     AX,RES_BIOS_FLAG        ;see if kybd flag set
02F4  25 0001                        AND     AX,1                    ;
02F7  74 07                          JZ      NOT_EGA_KBD             ;if not, go chk KBD_BUF
02F9  B8 0040                        MOV     AX,0040H                ;set DS = 40
02FC  8E D8                          MOV     DS,AX                   ;
02FE  EB 10                          JMP     SHORT CHECK_BUF         ;
```

```
0300  80 3E 05D7 R 00        NOT_EGA_KBD:  CMP     SCCFLAG,0               ;see if 7312 or 7314 installed
0305  74 09                                 JE      CHECK_BUF               ;if not, check bufr
0307  BE 0516 R                             MOV     SI,OFFSET SER1_DATA     ;
030A  8B 5C 08                              MOV     BX,[SI].BUFFER_HEAD     ;BX = SER1 bufr head
030D  8B 4C 0A                              MOV     CX,[SI].BUFFER_TAIL     ;AX = SER1 bufr tail
0310  8A 07                   CHECK_BUF:    MOV     AL,BYTE PTR[BX]         ;cmpr received char to ESC
0312  1F                                    POP     DS                      ;
0313  3B CB                                 CMP     CX,BX                   ;
0315  74 19                                 JE      XITCHK_ESC              ;if head = tail, clr flag & exit
0317  3C 1B                                 CMP     AL,ESC                  ;
0319  75 15                                 JNE     XITCHK_ESC              ;if char not ESC, clr flag & exit
031B  B0 01                                 MOV     AL,01H                  ;set flag & return
031D  A2 06A5 R               XIT_CHKESC:   MOV     ESC_FLAG,AL             ;set flag mailbox
0320  A1 041C R                             MOV     AX,KBD_BUF_TAIL         ;flush all key buffers
0323  A3 041A R                             MOV     KBD_BUF_HEAD,AX         ;
0326  BE 0516 R                             MOV     SI,OFFSET SER1_DATA     ;
0329  8B 44 0A                              MOV     AX,[SI].BUFFER_TAIL     ;
032C  89 44 08                              MOV     [SI].BUFFER_HEAD,AX     ;
032F  C3                                    RET                             ;

0330  B0 00                   XITCHK_ESC:   MOV     AL,00H                  ;
0332  EB E9                                 JMP     SHORT XIT_CHKESC

                              CHK_ESC       ENDP                            ;

                              ;*************************************************************************
                              ; MEMDISK_IO                                                            *
                              ;       This interface provides access to the memory disk drives.      *
                              ;       (512 byte sectors)                                             *
                              ;                                                                       *
                              ; Entry:      AH = operation                                            *
                              ;               AH = 0  reset (no action taken)                         *
                              ;               AH = 1  read status (bad command)                       *
                              ;               AH = 2  read sectors into memory                        *
                              ;               AH = 3  write to sectors with verify                    *
                              ;               AH = 4  verify (no action taken)                        *
                              ;             DL = drive number, 0-15 allowed                           *
                              ;             CX = first logical sector, maximum 1024                   *
                              ;             AL = number of sectors, maximum 256                       *
                              ;             ES:BX = transfer address                                  *

                              ; Exit:       AL = number of sectors transferred                        *
                              ;             CO = successful disk operation, AH = 0                    *
                              ;             C1 = failure, AH = status                                 *
                              ; Alters:     Flags, AX                                                 *
                              ; Calls:      none                                                      *
                              ; Stack:                                                                *
                              ;*************************************************************************

                                            ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
0334                          MEMDISK_IO    PROC    FAR                     ;
0334  FB                      MIO:          STI                             ;interrupts back on
0335  06                                    PUSH    ES                      ;save registers
0336  1E                                    PUSH    DS                      ;
0337  56                                    PUSH    SI                      ;
0338  57                                    PUSH    DI                      ;
0339  52                                    PUSH    DX                      ;
033A  51                                    PUSH    CX                      ;
033B  53                                    PUSH    BX                      ;
033C  80 FC 00                              CMP     AH,0                    ;
033F  74 1A                                 JE      OKRET                   ;0 = init, no action
0341  80 FC 01                              CMP     AH,1                    ;
0344  74 0F                                 JE      BC_RET                  ;1 = read status, bad command
0346  80 FC 02                              CMP     AH,2                    ;
0349  74 1D                                 JE      XFER                    ;2 = read from memory disk
034B  80 FC 03                              CMP     AH,3                    ;
034E  74 18                                 JE      XFER                    ;3 = write to memory disk
0350  80 FC 04                              CMP     AH,4                    ;
0353  74 06                                 JE      OKRET                   ;4 = verify, no action
0355  B4 01                   BC_RET:       MOV     AH,BAD_CMD              ;AH > 5, bad command
0357  F9                      ERRRET:       STC                             ;set carry for error
0358  EB 04 90                             JMP     MRET                    ;
035B  B4 00                   OKRET:        MOV     AH,0                    ;return for successful
035D  F8                                    CLC                             ;   operation
035E  5B                      MRET:         POP     BX                      ;restore registers
035F  59                                    POP     CX                      ;
```

```
0360  5A                              POP     DX                      ;
0361  5F                              POP     DI                      ;
0362  5E                              POP     SI                      ;
0363  1F                              POP     DS                      ;
0364  07                              POP     ES                      ;
0365  CA 0002                        RET     2                       ;throw away flags

0368  8B FB                   XFER:    MOV     DI,BX                   ;xfer addr to ES:DI
036A  8B D8                            MOV     BX,AX                   ;operation/no. sectors to BX

                                  ;----- set up port address:
036C  80 FA 10                         CMP     DL,16                   ;if drive no. > 15, return
036F  73 E4                            JAE     BC_RET                  ;   bad command
0371  32 F6                            XOR     DH,DH                   ;clear DH
0373  03 D2                            ADD     DX,DX                   ;form port address, this disk
0375  81 C2 0280                       ADD     DX,ROMBASEPORT          ;

                                  ;----- set up starting sector and disk memory addr for xfer:
0379  8B C1                            MOV     AX,CX                   ;starting sector to AX
037B  3D 0401                          CMP     AX,1025                 ;if sector > 1024, return
037E  73 D5                            JAE     BC_RET                  ;   bad command
0380  81 FA 0282                       CMP     DX,ROMBASEPORT+2        ;
0384  75 0D                            JNE     DISKNOTB                ;if not mem disk B, continue
0386  E8 03F1 R                        CALL    DDS                     ;
                                       ASSUME  DS:IODATA               ;set DS = iodata
0389  80 3E 06A2 R 00                  CMP     RAMDRVBFLG,0            ;
038E  74 03                            JE      DISKNOTB                ;if flag not set, continue
0390  80 CC 08                         OR      AH,08H                  ;add bit meaning 32k ram device
0393  B9 E000                 DISKNOTB: MOV     CX,DISKSEG              ;memory disk address to DS:SI
0396  8E D9                            MOV     DS,CX                   ;
                                       ASSUME  DS:NOTHING              ;
0398  BE 0000                          MOV     SI,0                    ;

039B  80 FF 02                         CMP     BH,2                    ;read operation?
039E  74 06                            JE      X1                      ;   -jump if so
03A0  06                               PUSH    ES                      ;exchange source and destination
03A1  1E                               PUSH    DS                      ; transfer addresses for write
03A2  07                               POP     ES                      ;
03A3  1F                               POP     DS                      ;
03A4  87 F7                            XCHG    SI,DI                   ;

                                  ;----- this is the actual data transfer and verify:
03A6  53                       X1:     PUSH    BX                      ;save # of sectors to xfer
03A7  80 CC 04                         OR      AH,004H                 ;set port enable bit
03AA  EE                       X2:     OUT     DX,AL                   ;enable memory disk
03AB  86 C4                            XCHG    AL,AH                   ;
03AD  42                               INC     DX                      ;
03AE  EE                               OUT     DX,AL                   ;
03AF  4A                               DEC     DX                      ;set DX = port again
03B0  86 C4                            XCHG    AL,AH                   ;set AX to enable disk again

03B2  B9 0200                          MOV     CX,512                  ;bytes/xfer: 512 bytes
03B5  06                               PUSH    ES                      ;preserve destination address
03B6  57                               PUSH    DI                      ;
03B7  1E                               PUSH    DS                      ;preserve source address
03B8  56                               PUSH    SI                      ;
03B9  F3/ A4                           REP     MOVSB                   ;transfer data
03BB  5E                               POP     SI                      ;restore source address
03BC  1F                               POP     DS                      ;
03BD  5F                               POP     DI                      ;restore destination address
03BE  07                               POP     ES                      ;
03BF  B9 0200                          MOV     CX,512                  ;restore transfer count
03C2  F3/ A6                           REP     CMPSB                   ;verify the data
03C4  0B C9                            OR      CX,CX                   ;test for all bytes transferred
03C6  75 14                            JNZ     X6                      ;exit if error

                                  ;----- more sectors to do?
03C8  40                               INC     AX                      ;point AX to next sector
03C9  FE CB                            DEC     BL                      ;more sectors?
03CB  74 0F                            JZ      X6                      ;   -jump if not
03CD  80 FF 02                         CMP     BH,2                    ;set "disk" offset to 0
03D0  75 05                            JNE     X3                      ;
03D2  BE 0000                          MOV     SI,0                    ;
03D5  EB 03                            JMP     X2                      ;
03D7  BF 0000                  X3:     MOV     DI,0                    ;
03DA  EB CE                            JMP     X2                      ;
```

```
                        ;----- finished transferring:
03DC  B0 00             X6:     MOV     AL,000H                 ;disable memory card
03DE  EE                          OUT     DX,AL                   ;
03DF  42                        INC     DX                      ;
03E0  EE                        OUT     DX,AL                   ;
03E1  58                        POP     AX                      ;get # sectors from stack
03E2  2A C3                     SUB     AL,BL                   ;set AL to # actually xfered
03E4  80 FB 00                  CMP     BL,0                    ;successful if BL = 0
03E7  74 05                     JE      X7                      ;
03E9  B4 03                     MOV     AH,WRITE_FAIL           ;error if BL >0
03EB  E9 0357 R                 JMP     ERRRET                  ;
03EE  E9 035B R         X7:     JMP     OKRET                   ;

                        MEMDISK_IO      ENDP                    ;

                                SUBTTL  BIOS Support Routines
                        ;*************************************************************************
                        ;* DDS                                                                   *
                        ;*             Sets DS to the IODATA segment                             *
                        ;*                                                                       *
                        ;* Entry:      none                                                      *
                        ;* Exit:       DS = IODATA                                               *
                        ;* Alters:     DS                                                        *
                        ;* Calls:      none                                                      *
                        ;* Stack:      02H words                                                 *
                        ;*************************************************************************
                                ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
03F1                    DDS     PROC    NEAR                    ;set DS to IODATA
03F1  50                        PUSH    AX                      ;
03F2  B8 ---- R                 MOV     AX,SEG IODATA           ;
03F5  8E D8                     MOV     DS,AX                   ;
03F7  58                        POP     AX                      ;
03F8  C3                        RET                             ;
                        DDS     ENDP                            ;

                        ;*************************************************************************
                        ;* CHECK_RAM                                                             *
                        ;*              Checks if a byte is RAM.                                 *
                        ;*                                                                       *
                        ;* Entry:       CX = segment address                                     *
                        ;* Exit:        Zero flag set if RAM                                     *
                        ;* Alters:      FLAGS                                                    *
                        ;* Calls:       none                                                     *
                        ;* Stack:       03H words                                                *
                        ;*************************************************************************
                                ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
03F9                    CHECK_RAM       PROC    NEAR            ;
03F9  1E                        PUSH    DS                      ;
03FA  50                        PUSH    AX                      ;
03FB  8E D9                     MOV     DS,CX                   ;
                                ASSUME  DS:NOTHING              ;DS = segment to check
03FD  A0 0000                   MOV     AL,DS:[0000]            ;get byte
0400  F6 D0                     NOT     AL                      ;complement byte
0402  A2 0000                   MOV     DS:[0000],AL            ;store complemented byte
0405  38 06 0000                CMP     DS:[0000],AL            ;check if write was successful
0409  F6 16 0000                NOT     BYTE PTR DS:[0000]      ;restore byte
040D  58                        POP     AX                      ;
040E  1F                        POP     DS                      ;
040F  C3                        RET                             ;
                        CHECK_RAM       ENDP                    ;

                        ;*************************************************************************
                        ;* FIND_RAM_TOP                                                          *
                        ;*              Searches memory for RAM starting at 0000:0000.  This is a*
                        ;*              non-destructive search.  Sets BP to the first            *
                        ;*              paragraph above top of RAM.                              *
                        ;*                                                                       *
                        ;* Entry:       none                                                     *
                        ;* Exit:        BP = size of contiguous RAM in paragraphs                *
                        ;* Alters:      FLAGS,CX,BP                                              *
                        ;* Calls:       CHECK_RAM                                                *
                        ;* Stack:       04H words                                                *
                        ;*************************************************************************
                                ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
0410                    FIND_RAM_TOP    PROC    NEAR            ;
```

```
0410  B9 FE00                MOV     CX,-0200H               ;start at 0000:0000
0413  81 C1 0200            CHK_RAM:ADD     CX,0200H                ;point to next 8K
0417  E8 03F9 R                    CALL    CHECK_RAM               ;see if CX:0000 is RAM
041A  74 F7                        JZ      CHK_RAM                 ;loop while RAM is found
041C  8B E9                        MOV     BP,CX                   ;move size (paragraphs) to BP
041E  8B C1                        MOV     AX,CX                   ;divide by 64 = size in Kbytes
0420  BA 0000                      MOV     DX,0                    ;
0423  B9 0040                      MOV     CX,64                   ;
0426  F7 F1                        DIV     CX                      ;
0428  83 FA 00                     CMP     DX,0                    ;
042B  74 01                        JE      XITRAMSIZE              ;
042D  40                           INC     AX                      ;
042E  A3 0413 R             XITRAMSIZE:MOV     MEM_SIZE,AX             ;store mem size in Kbytes
0431  C3                           RET                             ;
                            FIND_RAM_TOP ENDP                          ;

                            ;*************************************************************************
                            ;* FIND_DISK                                                             *
                            ;*              Checks memory for a memory disk ID string.               *
                            ;*                                                                       *
                            ;* Entry:       CX = segment to check                                    *
                            ;* Exit:        Zero set if no disk                                      *
                            ;*              Not Zero if disk found                                   *
                            ;* Alters:      FLAGS,AX,BX,DI,SI                                        *
                            ;* Calls:       none                                                     *
                            ;* Stack:       05H words                                                *
                            ;*************************************************************************
                                        ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
0432                        FIND_DISK   PROC    NEAR                    ;
0432  06                                PUSH    ES                      ;
0433  51                                PUSH    CX                      ;
0434  1E                                PUSH    DS                      ;
0435  0E                                PUSH    CS                      ;
0436  1F                                POP     DS                      ;
                                        ASSUME  DS:CODE                 ;
0437  8D 36 1814 R                      LEA     SI,MEMDISKID            ;DS:SI = id string
043B  8E C1                            MOV     ES,CX                   ;segment to check
                                        ASSUME  ES:NOTHING              ;
043D  BF 0003                          MOV     DI,BOOT_SECTOR.OEM_NAME ;ES:DI = RAM disk sector 0?
0440  B9 0008 90                       MOV     CX,IDSIZE               ;number of bytes in id
0444  F3/ A6                          REPE    CMPSB                   ;check for id
0446  1F                              POP     DS                      ;
                                        ASSUME  DS:IODATA               ;
0447  B0 FF                           MOV     AL,0FFH                 ;AL = OFF if disk found
0449  E3 02                           JCXZ    FOUND_DISK              ;jump if memory disk found
044B  32 C0                           XOR     AL,AL                   ;AL = 0 if no disk
044D  22 C0                 FOUND_DISK: AND     AL,AL                   ;set zero flag as appropriate
044F  59                              POP     CX                      ;
0450  07                              POP     ES                      ;
0451  C3                              RET                             ;
                            FIND_DISK   ENDP                            ;

                            ;*************************************************************************
                            ;* CLEAR_RAM                                                             *
                            ;*              Clears all RAM from 1000:0000 to (BP-1):FFFF             *
                            ;*                                                                       *
                            ;* Entry:       DS = IODATA                                              *
                            ;*              BP = size of contiguous RAM in paragraphs                *
                            ;* Exit:        none                                                     *
                            ;* Alters:      FLAGS,AX,CX,DX,DI                                        *
                            ;* Calls:       none                                                     *
                            ;* Stack:       02H words                                                *
                            ;*************************************************************************
                                        ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
0452                        CLEAR_RAM   PROC    NEAR                    ;
0452  06                                PUSH    ES                      ;
0453  8B D5                            MOV     DX,BP                   ;get memory size
0455  B8 0000                          MOV     AX,0                    ;going to clear RAM
0458  BF 0000                 INIT_MEMORY:MOV     DI,0                    ;start at offset 0000
045B  81 EA 1000                       SUB     DX,1000H                ;decrement down to the next segment
045F  81 FA 1000                       CMP     DX,1000H                ;last segment to initialize?
0463  72 09                            JB      LASTSEG                 ;  if last segment, jump
0465  8E C2                            MOV     ES,DX                   ;ES = segment to clear
                                        ASSUME  ES:NOTHING              ;
0467  B9 8000                          MOV     CX,8000H                ;32K words
046A  F3/ AB                           REP     STOSW                   ;clear 64K bytes
```

```
046C  EB EA                     JMP     INIT_MEMORY             ;
046E  83 FA 00          LASTSEG:CMP     DX,0                    ;
0471  74 12                     JE      CLR_RAM_RET             ;
0473  B9 1000                   MOV     CX,1000H                ;clear remaining memory
0476  2B CA                     SUB     CX,DX                   ;count in paragraphs to clear
0478  D1 E1                     SHL     CX,1                    ;convert paragraphs to words
047A  D1 E1                     SHL     CX,1                    ;
047C  D1 E1                     SHL     CX,1                    ;
047E  BA 1000                   MOV     DX,1000H                ;last segment to clear
0481  8E C2                     MOV     ES,DX                   ;
                                ASSUME  ES:NOTHING              ;
0483  F3/ AB                    REP     STOSW                   ;clear last 64K bytes or less
0485  07                CLR_RAM_RET:POP ES                      ;
0486  C3                        RET                             ;
                          CLEAR_RAM ENDP                        ;

                        ;*************************************************************************
                        ;* MESSAGE                                                               *
                        ;*             Send a message to the console.                            *
                        ;*                                                                       *
                        ;* Entry:     SI = offset in code segment of $ terminated string         *
                        ;* Exit:      none                                                       *
                        ;* Alters:    FLAGS                                                      *
                        ;* Calls:     INT 29H                                                    *
                        ;* Stack:     15H words                                                  *
                        ;*************************************************************************
                                ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
0487                    MESSAGE PROC    NEAR                    ;
0487  55                       PUSH    BP                      ;
0488  56                       PUSH    SI                      ;
0489  50                       PUSH    AX                      ;
048A  2E: AC            NEXT_CHAR:LODS  CS:DISKERROR            ;get the character
048C  3C 24                     CMP     AL,'$'                  ;check for end of string
048E  74 04                     JE      MESSAGE_RET             ;
0490  CD 29                     INT     29H                     ;output the character
0492  EB F6                     JMP     NEXT_CHAR               ;
0494  58               MESSAGE_RET:POP   AX                      ;
0495  5E                       POP     SI                      ;
0496  5D                       POP     BP                      ;
0497  C3                       RET                             ;
                        MESSAGE ENDP                            ;

                        ;*************************************************************************
                        ;* HEX_OUT_BYTE                                                          *
                        ;*             Outputs a byte in hexadecimal                             *
                        ;*                                                                       *
                        ;* Entry:     AL = byte to output                                        *
                        ;* Exit:      none                                                       *
                        ;* Alters:    FLAGS                                                      *
                        ;* Calls:     INT 29H                                                    *
                        ;* Stack:     15H words                                                  *
                        ;*************************************************************************
                                ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
0498                    HEX_OUT_BYTE PROC NEAR                  ;
0498  50                       PUSH    AX                      ;save ax
0499  8A E0                     MOV     AH,AL                   ;save al
049B  D0 E8                     SHR     AL,1                    ;get most significant nibble of al
049D  D0 E8                     SHR     AL,1                    ;
049F  D0 E8                     SHR     AL,1                    ;
04A1  D0 E8                     SHR     AL,1                    ;
04A3  E8 04AF R                 CALL    HEX_OUT_NIBL
04A6  8A C4                     MOV     AL,AH
04A8  24 0F                     AND     AL,0FH
04AA  E8 04AF R                 CALL    HEX_OUT_NIBL
04AD  58                        POP     AX
04AE  C3                        RET                             ;restore ax
                        HEX_OUT_BYTE ENDP                       ;return from hex al

04AF                    HEX_OUT_NIBL PROC NEAR
04AF  1E                        PUSH    DS
04B0  53                        PUSH    BX
04B1  0E                        PUSH    CS
04B2  1F                        POP     DS
                                ASSUME  DS:CODE
04B3  8D 1E 04BD R              LEA     BX,HEX_TABLE
04B7  D7                        XLAT
```

```
04B8  CD 29                     INT     29H
04BA  5B                        POP     BX
04BB  1F                        POP     DS
04BC  C3                        RET

04BD  30 31 32 33 34 35 36      HEX_TABLE:      DB      '0123456789ABCDEF'      ;hex to ascii table
      37 38 39 41 42 43 44
      45 46

                                HEX_OUT_NIBL    ENDP                            ;

                                ;*****************************************************************************
                                ;* HEX_OUT_WORD                                                              *
                                ;*              Outputs a word in hexadecimal                                *
                                ;*                                                                           *
                                ;*                                                                           *
                                ;* Entry:       AX = word to output                                          *
                                ;* Exit:        none                                                         *
                                ;* Alters:      none                                                         *
                                ;* Calls:       HEX_OUT_BYTE                                                 *
                                ;* Stack:       17H words                                                    *
                                ;*****************************************************************************
                                ;
                                        ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
04CD                            HEX_OUT_WORD    PROC    NEAR                    ;
04CD  9C                                        PUSHF                           ;
04CE  86 C4                                     XCHG    AL,AH                   ;
04D0  E8 0498 R                                 CALL    HEX_OUT_BYTE            ;output high byte
04D3  86 C4                                     XCHG    AL,AH                   ;
04D5  E8 0498 R                                 CALL    HEX_OUT_BYTE            ;output low byte
04D8  9D                                        POPF                            ;
04D9  C3                                        RET                             ;
                                HEX_OUT_WORD    ENDP                            ;

                                        SUBTTL  SCC Initialization

                                ;*****************************************************************************
                                ;* SCCINIT                                                                   *
                                ;*              Checks system for SCC devices on 7312 or 7314 cards.         *
                                ;*              Devices found and corresponding IODATA areas are initalized  *
                                ;*              as MS-DOS character devices depending on which cards are     *
                                ;*              installed in system:                                         *
                                ;*              7312: Ch.A = SER1, Ch.B = SER2                               *
                                ;*              7314: Ch.A = SER1, Ch.B = SER2, Ch.C = SER3, Ch.D = SER4     *
                                ;*                                                                           *
                                ;*              SCC interupts are enabled in IO.SYS, REINIT                  *
                                ;*              A mailbox "SCCFLAG" in IODATA will be set = # of SCC's       *
                                ;*              (NONE = 0, 7312 = 2, 7314 = 4)                               *
                                ;* Entry:       none                                                         *
                                ;* Exit:        none                                                         *
                                ;* Alters:      FLAGS                                                        *
                                ;* Calls:       SYSCHECK, INIT2SCC, INIT4SCC, INIT, BUFINIT                  *
                                ;* Stack:       11H words                                                    *
                                ;*****************************************************************************
                                ;
                                        ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
04DA                            SCCINIT         PROC    NEAR                    ;
04DA  55                                        PUSH    BP                      ;
04DB  06                                        PUSH    ES                      ;save registers
04DC  1E                                        PUSH    DS                      ;
04DD  53                                        PUSH    BX                      ;
04DE  E8 03F1 R                                 CALL    DDS                     ;set DS = IODATA
                                                ASSUME  DS:IODATA               ;
04E1  E8 0525 R                                 CALL    CHKSCC                  ;set BL = SCC's code
04E4  88 1E 05D7 R                              MOV     SCCFLAG,BL              ;store SCC flag
04E8  80 FB 00                                  CMP     BL,00H                  ;BL shows devices in system
04EB  74 07                                     JE      XITSCCINIT              ;if no SCC's, exit
04ED  80 FB 02                                  CMP     BL,02H                  ;
04F0  74 1D                                     JE      SCC2                    ;if 7312 only...
04F2  EB 29                                     JMP     SHORT INIT4SCC          ;init SER1 thru SER4

04F4  33 DB                     XITSCCINIT:     XOR     BX,BX                   ;set DS = vect tbl segment
04F6  8E DB                                     MOV     DS,BX                   ;
                                                ASSUME  DS:NOTHING              ;
04F8  BF 01C4 R                                 MOV     DI,OFFSET VECTSCC_TRAP  ;tbl entry for common SCC bios routine
04FB  BB 12CB R                                 MOV     BX,OFFSET SCCBIOS       ;
04FE  E8 12C3 R                                 CALL    MOVEADDR                ;
0501  BF 00A4 R                                 MOV     DI,OFFSET TTY_VECT      ;table entry for INT 29H write TTY routine
0504  BB 14B6 R                                 MOV     BX,OFFSET WRITE_TTY     ;
0507  E8 12C3 R                                 CALL    MOVEADDR                ;
```

```
050A  5B                  EXITINITSCC:  POP     BX                      ;
050B  1F                                POP     DS                      ;restore and return
050C  07                                POP     ES                      ;
050D  5D                                POP     BP                      ;
050E  C3                                RET                             ;

050F  E8 1123 R           SCC2:         CALL    INIT2SCC                ;A & B of 7312 = SER1 & SER2
0512  E8 110D R           XIT2SCC:      CALL    INITSCC                 ;init selected SCC devices
0515  E8 123F R                         CALL    BUFINIT                 ;init SCC buffer pointers
0518  E8 058F R                         CALL    ENABLE_SCC
051B  EB ED                             JMP     SHORT EXITINITSCC       ;exit

051D  E8 1123 R           INIT4SCC:     CALL    INIT2SCC                ;A & B of 7314 = SER1 & SER2
0520  E8 1189 R                         CALL    INITMORSCC              ;C & D of 7314 = SER3 & SER4
0523  EB ED                             JMP     SHORT XIT2SCC           ;go initialize & exit

                          SCCINIT       ENDP                            ;

                          ;*************************************************************************
                          ;* CHKSCC                                                                *
                          ;*           This routine checks to see which SCC's (if any) are         *
                          ;*           installed in system                                         *
                          ;* Entry:    none                                                        *
                          ;* Exit:     BL indicates which SCCs are installed:                      *
                          ;*           BL=00=none, BL=02=7312, BL=04=7314                          *
                          ;* Alters:   BL,AX,FLAGS                                                 *
                          ;* Calls:    CHECK_SCC                                                   *
                          ;* Stack:    02H words                                                   *
                          ;*************************************************************************
                                    ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
0525                      CHKSCC    PROC    NEAR                    ;check for SCCs in system
0525  52                          PUSH    DX                      ;
0526  B3 00                       MOV     BL,0                    ;init BL = 0 = no scc
0528  BA 00B1                     MOV     DX,CHA14                ;check for 7314 SCC ch A
052B  B4 80                       MOV     AH,VECT14A              ;
052D  E8 0542 R                   CALL    CHECKSCC                ;init & chk for SCC A
0530  75 02                       JNZ     CHKSCC3                 ;if not found, skip
0532  B3 02                       MOV     BL,2                    ;set BL = 2 = scc A & B found
0534  BA 00B5             CHKSCC3:       MOV     DX,CHC14                ;check for 7314 SCC ch C
0537  B4 81                       MOV     AH,VECT14C              ;
0539  E8 0542 R                   CALL    CHECKSCC                ;
053C  75 02                       JNZ     XITCHKSCC               ;if not found, skip
053E  B3 04                       MOV     BL,04H                  ;set BL = 4 = A,B,C & D found
0540  5A                  XITCHKSCC:     POP     DX                      ;
0541  C3                           RET                             ;
                          CHKSCC    ENDP                            ;

                          ;*************************************************************************
                          ;* CHECKSCC                                                              *
                          ;*           Checks to see if interrupt vector in AH can be written to   *
                          ;*           SCC device at port address DX (i.e., is SCC there?)         *
                          ;*                                                                       *
                          ;* Entry:    DX=port address, AH=interrupt vector                        *
                          ;* Exit:     ZF=1, SCC device is installed in system                     *
                          ;*           ZF=0, no SCC device at this address                         *
                          ;* Alters:   AL, FLAGS                                                   *
                          ;* Calls:    none                                                        *
                          ;* Stack:    01H word                                                    *
                          ;*************************************************************************
                                    ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
0542                      CHECKSCC  PROC    NEAR                    ;DX=port, AH=int vector
0542  B0 09                       MOV     AL,9                    ;
                                  SCCOUT                          ;
0544  EE                  1               OUT     DX,AL                   ;SCC recovery
0545  87 D2               1               XCHG    DX,DX                   ;
0547  87 D2               1               XCHG    DX,DX                   ;
0549  B0 C0                       MOV     AL,0C0H                 ;force hardware reset to the SCC
                                  SCCOUT                          ;
054B  EE                  1               OUT     DX,AL                   ;SCC recovery
054C  87 D2               1               XCHG    DX,DX                   ;
054E  87 D2               1               XCHG    DX,DX                   ;
0550  87 D2                       XCHG    DX,DX                   ;extra delay for reset
0552  B0 09                       MOV     AL,9                    ;
                                  SCCOUT                          ;
0554  EE                  1               OUT     DX,AL                   ;SCC recovery
0555  87 D2               1               XCHG    DX,DX                   ;
```

```
0557  87 D2                1              XCHG    DX,DX                   ;
0559  B0 80                               MOV     AL,080H                 ;reset SCC chan A
                                          SCCOUT                          ;
055B  EE                   1              OUT     DX,AL                   ;SCC recovery
055C  87 D2                1              XCHG    DX,DX                   ;
055E  87 D2                1              XCHG    DX,DX                   ;
0560  87 D2                               XCHG    DX,DX                   ;extra delay for reset
0562  B0 09                               MOV     AL,9                    ;
                                          SCCOUT                          ;
0564  EE                   1              OUT     DX,AL                   ;SCC recovery
0565  87 D2                1              XCHG    DX,DX                   ;
0567  87 D2                1              XCHG    DX,DX                   ;
0569  B0 40                               MOV     AL,040H                 ;reset SCC chan B
                                          SCCOUT                          ;
056B  EE                   1              OUT     DX,AL                   ;SCC recovery
056C  87 D2                1              XCHG    DX,DX                   ;
056E  87 D2                1              XCHG    DX,DX                   ;
0570  87 D2                               XCHG    DX,DX                   ;extra delay for reset
0572  B0 02                               MOV     AL,2                    ;write int vector
                                          SCCOUT                          ;
0574  EE                   1              OUT     DX,AL                   ;SCC recovery
0575  87 D2                1              XCHG    DX,DX                   ;
0577  87 D2                1              XCHG    DX,DX                   ;
0579  8A C4                               MOV     AL,AH                   ;
                                          SCCOUT                          ;
057B  EE                   1              OUT     DX,AL                   ;SCC recovery
057C  87 D2                1              XCHG    DX,DX                   ;
057E  87 D2                1              XCHG    DX,DX                   ;
0580  B0 02                               MOV     AL,2                    ;read interrupt vector back in
                                          SCCOUT                          ;
0582  EE                   1              OUT     DX,AL                   ;SCC recovery
0583  87 D2                1              XCHG    DX,DX                   ;
0585  87 D2                1              XCHG    DX,DX                   ;
                                          SCCIN                           ;
0587  EC                   1              IN      AL,DX                   ;SCC recovery
0588  87 D2                1              XCHG    DX,DX                   ;
058A  87 D2                1              XCHG    DX,DX                   ;
058C  3A C4                               CMP     AL,AH                   ;same as written?
058E  C3                                  RET                             ;if so ZF=1
                                  CHECKSCC        ENDP                    ;

                                  ;*****************************************************************************
                                  ;* ENABLE_SCC                                                                *
                                  ;*       The SCC interupts are enabled.                                      *
                                  ;* Entry:  none                                                              *
                                  ;* Exit:   none                                                              *
                                  ;* Alters: none                                                              *
                                  ;* Calls:  none                                                              *
                                  ;* Stack:  8 words                                                           *
                                  ;*****************************************************************************
                                          ASSUME  CS:CODE, DS:NOTHING, ES:NOTHING
058F                              ENABLE_SCC      PROC    NEAR            ;
058F  50                                  PUSH    AX                      ;console SCC interrupt enable
0590  53                                  PUSH    BX                      ;
0591  51                                  PUSH    CX                      ;
0592  57                                  PUSH    DI                      ;
0593  06                                  PUSH    ES                      ;
0594  0E                                  PUSH    CS                      ;set ES=CS
0595  07                                  POP     ES                      ;
0596  BB 0001                             MOV     BX,SER2_DEV_NUM         ;load SCC device number
0599  BF 0587 R                           MOV     DI,OFFSET TABLE         ;load offset of initialization table
059C  B9 0002                             MOV     CX,2                    ;number of bytes to CX
059F  B4 00                               MOV     AH,SCC_INIT             ;
05A1  CD 71                               INT     SCC_TRAP                ;
05A3  BB 0003                             MOV     BX,SER4_DEV_NUM         ;load SCC device number
05A6  BF 0587 R                           MOV     DI,OFFSET TABLE         ;load offset of initialization table
05A9  B9 0002                             MOV     CX,2                    ;number of bytes to CX
05AC  B4 00                               MOV     AH,SCC_INIT             ;
05AE  CD 71                               INT     SCC_TRAP                ;
05B0  07                                  POP     ES                      ;
05B1  5F                                  POP     DI                      ;
05B2  59                                  POP     CX                      ;
05B3  5B                                  POP     BX                      ;
05B4  58                                  POP     AX                      ;
05B5  FB                                  STI                             ;enable interrupts
05B6  C3                                  RET                             ;
```

```
05B7  09 09                    TABLE       DB      9,009H                  ;WR9, master interrupt enable
                               ENABLE_SCC  ENDP

                               ;*********************************************************************
                               ;* CHKCONSOLE                                                        *
                               ;*          If EGA flag set, do nothing; If no SCC, set COM1 for CON *
                               ;* ENTRY:   none                                                     *
                               ;* EXIT:    none                                                     *
                               ;* ALTERS:  all                                                      *
                               ;* CALLS:   none                                                     *
                               ;* STACK:   4 words                                                  *
                               ;*********************************************************************
05B9                           CHKCONSOLE  PROC    NEAR                    ;
05B9  E8 03F1 R                            CALL    DDS                     ;set DS = iodata
                                           ASSUME  DS:IODATA               ;
05BC  A1 048B R                            MOV     AX,RES_BIOS_FLAG        ;
05BF  24 01                                AND     AL,01H                  ;
05C1  75 0B                                JNZ     XITCHKCONSOLE           ;if EGA/KBD installed, exit
05C3  A0 05D7 R                            MOV     AL,SCCFLAG              ;
05C6  3C 00                                CMP     AL,0                    ;
05C8  75 04                                JNE     XITCHKCONSOLE           ;if 7312 or 7314 installed, exit
05CA  E8 05CF R                            CALL    COM1TOCON               ;init 8250 & 8259 for COM1 = CON
05CD  C3                                   RET                             ;
05CE  C3                       XITCHKCONSOLE:RET                           ;
                               CHKCONSOLE  ENDP                            ;

                               ;------This routine will set the COM1 uart to 19.2k baud, enable the
                               ;       8250 recv char interupt and program the 8259 for kybd interupt (int C).

= 00EF                         INTMASK_8250  EQU     0EFH

05CF                                       ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
05CF  E8 05E9 R                COM1TOCON   PROC    NEAR                    ;
05D2  E8 0615 R                            CALL    REINIT_8250             ;re-init 8250 to generate interrupt & 19.2k baud
05D5  C7 06 041A R 041E R                  CALL    REINIT_8259             ;re-init 8259 to generate int 0C when key on COM1
05DB  C7 06 041C R 041E R                  MOV     KBD_BUF_HEAD,OFFSET KBD_BUFFER ;
05E1  BE 0516 R                            MOV     KBD_BUF_TAIL,OFFSET KBD_BUFFER ;
05E4  C6 44 0D 01                          MOV     SI,OFFSET SER1_DATA     ;set handshake on for ser1
05E8  C3                                   MOV     [SI].HANDSHAKE,1        ;
                                           RET                             ;
                               COM1TOCON   ENDP                            ;

                               ;------REINIT_8250 initializes the 8250 uart chip for 19.2K and enables interupts.
                                           ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
05E9                           REINIT_8250 PROC    NEAR                    ;
05E9  52                                   PUSH    DX                      ;
05EA  BA 03F8                              MOV     DX,BASE_8250            ;set DX = 8250 base adr
05ED  83 C2 03                             ADD     DX,3                    ;point to line control
05F0  EC                                   IN      AL,DX                   ;
05F1  0C 80                                OR      AL,80H                  ;enable divisor latch registers
05F3  EE                                   OUT     DX,AL                   ;
05F4  50                                   PUSH    AX                      ;save line status
05F5  83 EA 03                             SUB     DX,3                    ;point to divisor latch 0
05F8  B8 0006                              MOV     AX,0006H                ;set divisor = 19.2K baud
05FB  EE                                   OUT     DX,AL                   ;
05FC  42                                   INC     DX                      ;
05FD  8A C4                                MOV     AL,AH                   ;
05FF  EE                                   OUT     DX,AL                   ;
0600  83 C2 02                             ADD     DX,2                    ;point back to line control
0603  58                                   POP     AX                      ;
0604  24 7F                                AND     AL,7FH                  ;disable divisor lathh registers
0606  EE                                   OUT     DX,AL                   ;
0607  83 EA 02                             SUB     DX,2                    ;point to interupt enable register
060A  B0 01                                MOV     AL,01H                  ;
060C  EE                                   OUT     DX,AL                   ;enable recv char available int
060D  83 C2 03                             ADD     DX,3                    ;point to modem cntl reg
0610  B0 0F                                MOV     AL,0FH                  ;
0612  EE                                   OUT     DX,AL                   ;enable handshake & outputs
0613  5A                                   POP     DX                      ;
0614  C3                                   RET                             ;
                               REINIT_8250 ENDP                            ;

                               ;------REINIT_8259 initializes the interupt controller chip to include
                               ;       the 0C interupt for the 8250 interupt.
                                           ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
0615                           REINIT_8259 PROC    NEAR                    ;
```

```
0615  FA                        CLI                             ;interupts off
0616  B0 13                     MOV     AL,13H                  ;edge trig, 8 interupts,
0618  E6 20                     OUT     PORTCNTL_8259,AL        ;single mode, icw4 needed
061A  B0 08                     MOV     AL,08H                  ;start at vect 8
061C  E6 21                     OUT     PORTDATA_8259,AL        ;
061E  B0 09                     MOV     AL,09H                  ;buffered 8088 mode
0620  E6 21                     OUT     PORTDATA_8259,AL        ;
0622  B0 FE                     MOV     AL,INTERUPT_MASK        ;get int mask info
0624  24 EF                     AND     AL,INTMASK_8250         ;add int C to mask
0626  E6 21                     OUT     PORTDATA_8259,AL        ;
0628  E4 21                     IN      AL,PORTDATA_8259        ;
062A  FB                        STI                             ;interupts back on
062B  C3                        RET                             ;
                      REINIT_8259   ENDP                        ;

                      ;*************************************************************************
                      ;* INIT_MEMSIZE                                                          *
                      ;*           Sets MEM_SIZE mailbox in low ram = # kbytes                 *
                      ;* ENTRY:  BP = mem size in paragraphs                                   *
                      ;* EXIT:   none                                                          *
                      ;* ALTERS: none                                                          *
                      ;* CALLS:  none                                                          *
                      ;* STACK:  2 words                                                       *
                      ;*************************************************************************
                                      ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
062C                  INIT_MEMSIZE  PROC    NEAR              ;
062C  52                            PUSH    DX                ;
062D  51                            PUSH    CX                ;save registers
062E  B9 0040                       MOV     CX,64             ;
0631  8B C5                         MOV     AX,BP             ;divide # paragraphs by 64
0633  BA 0000                       MOV     DX,0              ;
0636  F7 F1                         DIV     CX                ;
0638  0B D2                         OR      DX,DX             ;
063A  74 01                         JZ      EVENKBYTE         ;if no remainder, skip
063C  40                            INC     AX                ;round off to next kbyte
063D  A3 0413 R       EVENKBYTE:    MOV     MEM_SIZE,AX       ;store MEM_SIZE
0640  59                            POP     CX                ;
0641  5A                            POP     DX                ;
0642  C3                            RET                       ;
                      INIT_MEMSIZE  ENDP                        ;

                      ;**** INT 12 ***********************************************************
                      ;* GET MEM SIZE                                                          *
                      ;*           This interrupt routine will report to the caller the        *
                      ;*           size of available memory in 1k blocks. The size was         *
                      ;*           determined in "FIND_RAM_TOP" and set in "INIT_MEMSIZE".     *
                      ;* ENTRY:    none                                                        *
                      ;* EXIT:     AX = mem size in 1k blocks (contigious)                     *
                      ;* ALTERS:   AX                                                          *
                      ;*************************************************************************
                                      ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
0643                  GETMEMSIZE    PROC    FAR               ;
0643  FB                            STI                       ;turn interrupts back on
0644  1E                            PUSH    DS                ;
0645  E8 03F1 R                     CALL    DDS               ;set DS = IODATA
                                      ASSUME  DS:IODATA         ;
0648  A1 0413 R                     MOV     AX,MEM_SIZE       ;get mem size to AX
064B  1F                            POP     DS                ;
064C  CF                            IRET                      ;
                      GETMEMSIZE    ENDP                      ;

                      ;**** INT 11 ***********************************************************
                      ;* GET EQUIPMENT                                                         *
                      ;*           This interrupt routine will report to the caller the        *
                      ;*           EQUIP_FLAG which describes the optional periphials          *
                      ;*           installed in the system.                                    *
                      ;* ENTRY:    none                                                        *
                      ;* EXIT:     AX = bit coded equipment flag                               *
                      ;*           Bits 15,14   = # printers connected                         *
                      ;*           Bits 13,12   = not used                                     *
                      ;*           Bits 11,10,9 = # RS232 cards installed                      *
                      ;*           Bit  8       = not used                                     *
                      ;*           Bits 7,6     = # disk drives installed                      *
                      ;*           Bits 5,4     = video mode                                   *
                      ;*           Bits 3,2     = not used                                     *
                      ;*           Bit  1       = math coprocessor                             *
```

```
                        ;*          Bit  0       = IPL disk installed                    *
                        ;* ALTERS:   AX                                                 *
                        ;*****************************************************************
                        ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
064D                    GETEQUIP    PROC    FAR                 ;
064D  FB                            STI                         ;turn interrupts back on
064E  1E                            PUSH    DS                  ;
064F  E8 03F1 R                     CALL    DDS                 ;set DS = IODATA
                                    ASSUME  DS:IODATA           ;
0652  A1 0410 R                     MOV     AX,EQUIP_FLAG       ;get EQUIP FLAG to AX
0655  1F                            POP     DS                  ;
0656  CF                            IRET                        ;
                        GETEQUIP    ENDP                        ;

                        ;*****************************************************************
                        ;* INIT_HDWR                                                    *
                        ;*          Initializes 7888 CPU interupt chip (8259), counter/timer *
                        ;*          chip (8254), uart chip (8250) and initializes clock & tick *
                        ;*          interupt vectors.                                   *
                        ;* Entry:   none                                                *
                        ;* Exit:    none                                                *
                        ;* Alters:  none                                                *
                        ;* Calls:   MOVEADDR                                            *
                        ;* Stack:   3 words                                             *
                        ;*****************************************************************
                        ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
0657                    INIT_HDWR   PROC    NEAR                ;
0657  E8 0672 R                    CALL    VECTOR_INIT         ;init hardware & clock int vectors
065A  E8 0710 R                     CALL    INIT_8254           ;init 8254 clock/timer chip
065D  E8 06FB R                     CALL    INIT_8259           ;init 8259 int handler chip
0660  E8 072D R                     CALL    INITCOMS            ;init 8250 uart chip(s)
0663  E8 077F R                     CALL    INITPRNS            ;init printer ports
0666  E8 07CC R                     CALL    SET_EQUIP           ;set joystick and ser prntr bits
0669  E8 0800 R                     CALL    INITRAMDISK         ;find & format any ram disks
066C  C6 06 05D8 R 01               MOV     CON_RESET_ENABLE,01H ;enable con reset(alt f5)
0671  C3                            RET                         ;
                        INIT_HDWR   ENDP

                        ;------VECTOR_INIT initializes the interupt vectors for the real time clock
                        ;       (int 1AH), the tick counter (int 8H) and a dummy return for timer
                        ;       break (int 1CH). Also init the interupt vectors for SCC and TTY write.

0672                    VECTOR_INIT PROC    NEAR
0672  1E                           PUSH    DS                  ;save registers
0673  57                           PUSH    DI                  ;
0674  53                           PUSH    BX                  ;
0675  33 DB                        XOR     BX,BX               ;
0677  8E DB                        MOV     DS,BX               ;
                                    ASSUME  DS:NOTHING          ;set DS = low memory
0679  BF 0020 R                     MOV     DI,OFFSET TICK_TRAP ;
067C  BB 0C2C R                     MOV     BX,OFFSET TICK_INT  ;
067F  E8 12C3 R                     CALL    MOVEADDR            ;set TICK_INT vector
0682  BF 0068 R                     MOV     DI,OFFSET CLOCK_TRAP ;
0685  BB 0ECA R                     MOV     BX,OFFSET CLOCK_INT ;
0688  E8 12C3 R                     CALL    MOVEADDR            ;set CLOCK_INT vector
068B  BF 0070 R                     MOV     DI,OFFSET TIMER_BREAK ;
068E  BB 06FA R                     MOV     BX,OFFSET DUMMY_IRET ;
0691  E8 12C3 R                     CALL    MOVEADDR            ;set TIMER_BREAK int vector to dummy return
0694  BF 0050 R                     MOV     DI,OFFSET COM1_VECT ;init vector for 8250 port
0697  BB 0C95 R                     MOV     BX,OFFSET COM1_INT  ;
069A  E8 12C3 R                     CALL    MOVEADDR            ;
069D  BF 0030 R                     MOV     DI,OFFSET COM1IN_TRAP ;set COM1 input char interupt
06A0  BB 0D9F R                     MOV     BX,OFFSET COM1IN_INT ;
06A3  E8 12C3 R                     CALL    MOVEADDR            ;
06A6  BF 0044 R                     MOV     DI,OFFSET EQUIP_VECT ;set get equip flag vector
06A9  BB 064D R                     MOV     BX,OFFSET GETEQUIP  ;
06AC  E8 12C3 R                     CALL    MOVEADDR            ;
06AF  BF 0048 R                     MOV     DI,OFFSET MEMSIZE_VECT ;set get mem size vector
06B2  BB 0643 R                     MOV     BX,OFFSET GETMEMSIZE ;
06B5  E8 12C3 R                     CALL    MOVEADDR            ;
06B8  BF 0064 R                     MOV     DI,OFFSET BOOT_VECT ;set boot vector
06BB  BB 0285 R                     MOV     BX,OFFSET LOADBOOTSECT ;
06BE  E8 12C3 R                     CALL    MOVEADDR            ;
06C1  BF 005C R                     MOV     DI,OFFSET PRINT_VECT ;set printer vector
06C4  BB 0E55 R                     MOV     BX,OFFSET PRINT_INT ;
06C7  E8 12C3 R                     CALL    MOVEADDR            ;
```

```
06CA  BF 006C R                 MOV     DI,OFFSET KBD_BREAK     ;IRET for kybd break
06CD  BB 06FA R                 MOV     BX,OFFSET DUMMY_IRET    ;
06D0  E8 12C3 R                 CALL    MOVEADDR                ;
06D3  BF 0014 R                 MOV     DI,OFFSET PRNT_SCRN     ;print screen service
06D6  BB 1A80 R                 MOV     BX,OFFSET PRINT_SCREEN  ;
06D9  E8 12C3 R                 CALL    MOVEADDR                ;

06DC  BF 0074 R                 MOV     DI,OFFSET VIDEO_INIT    ;set vector 1D to video param tbl
06DF  BB 70A4 R                 MOV     BX,OFFSET VIDEO_PARAMS  ;
06E2  E8 12C3 R                 CALL    MOVEADDR                ;
06E5  51                        PUSH    CX                      ;this routine will set vectors
06E6  BB 0078 R                 MOV     BX,OFFSET DISK_PTR      ;1E and 1F to 0000:0000
06E9  B8 0000                   MOV     AX,0                    ;
06EC  B9 0004                   MOV     CX,4                    ;
06EF  89 07                     ZEROSLOOP:  MOV     WORD PTR[BX],AX     ;
06F1  43                        INC     BX                      ;
06F2  43                        INC     BX                      ;
06F3  E2 FA                     LOOP    ZEROSLOOP               ;
06F5  59                        POP     CX                      ;
06F6  5B                        POP     BX                      ;restore registers & exit
06F7  5F                        POP     DI                      ;
06F8  1F                        POP     DS                      ;
06F9  C3                        RET                             ;

06FA  CF                    DUMMY_IRET:     IRET                ;dummy IRET interupt vector

                            VECTOR_INIT     ENDP                ;

                            ;------INIT_8259 initializes the interupt controller chip

06FB                        INIT_8259       PROC    NEAR        ;
06FB  FA                                    CLI                 ;interupts off
06FC  B0 13                                 MOV     AL,13H      ;edge trig, 8 interupts,
06FE  E6 20                                 OUT     PORTCNTL_8259,AL    ;single mode, icw4 needed
0700  B0 08                                 MOV     AL,08H      ;start at vect 8
0702  E6 21                                 OUT     PORTDATA_8259,AL    ;
0704  B0 09                                 MOV     AL,09H      ;buffered 8088 mode
0706  E6 21                                 OUT     PORTDATA_8259,AL    ;
0708  B0 FE                                 MOV     AL,INTERUPT_MASK    ;int mask info
070A  E6 21                                 OUT     PORTDATA_8259,AL    ;
070C  E4 21                                 IN      AL,PORTDATA_8259    ;
070E  FB                                    STI                 ;interupts back on
070F  C3                                    RET                 ;
                            INIT_8259       ENDP                ;

                            ;------INIT_8254 initializes one channel of the timer/counter chip to
                            ;       count for the tick counter (approx 18.2 times per second)

0710                        INIT_8254       PROC    NEAR        ;
0710  B0 36                                 MOV     AL,CNTR0_INIT       ;init counter 0 = 65536
0712  E6 43                                 OUT     PORTCNTL_8254,AL    ;
0714  B0 00                                 MOV     AL,0        ;
0716  E6 40                                 OUT     PORT0_8254,AL       ;
0718  B0 00                                 MOV     AL,0        ;
071A  E6 40                                 OUT     PORT0_8254,AL       ;
071C  B0 68                                 MOV     AL,CNTR1_INIT       ;init cntr 1 = nothing
071E  E6 43                                 OUT     PORTCNTL_8254,AL    ;
0720  B0 B6                                 MOV     AL,CNTR2_INIT       ;init counter 2 = tone (?)
0722  E6 43                                 OUT     PORTCNTL_8254,AL    ;
0724  B0 00                                 MOV     AL,0        ;
0726  E6 42                                 OUT     PORT2_8254,AL       ;
0728  B0 08                                 MOV     AL,8        ;
072A  E6 42                                 OUT     PORT2_8254,AL       ;
072C  C3                                    RET                 ;
                            INIT_8254       ENDP                ;

                            ;------INITCOMS initializes any PC compatible rs232 ports(8250)

072D                        INITCOMS        PROC    NEAR        ;
072D  E8 03F1 R                             CALL    DDS         ;set DS = IODATA
                                            ASSUME  DS:IODATA   ;
0730  BF 0779 R                             MOV     DI,OFFSET TBL_COMPORT   ;set SI = com port adr table
0733  BE 0400 R                             MOV     SI,OFFSET RS232_BASE    ;set DI = RS232 port base
0736  B1 00                                 MOV     CL,0        ;init CL = device count
0738  BB 0000                               MOV     BX,0        ;init BX = 0 (port #)
073B  2E: 8B 15             INITCOMSLOOP:  MOV     DX,CS:[DI]  ;get port adr from tbl
```

```
073E  83 FA 00                    CMP     DX,0                    ;
0741  74 1C                       JE      XITINITCOMS             ;if end of table, exit
0743  42                          INC     DX                      ;adj DX to int id register
0744  42                          INC     DX                      ;
0745  EC                          IN      AL,DX                   ;input int id register
0746  A8 F8                       TEST    AL,0F8H                 ;
0748  75 0E                       JNZ     NOCOM                   ;if bits 3-7 not 0, dont init
074A  FE C1                       INC     CL                      ;incr device count
074C  4A                          DEC     DX                      ;adj DX back to base port adr
074D  4A                          DEC     DX                      ;
074E  89 14                       MOV     [SI],DX                 ;store port adr
0750  8B D3                       MOV     DX,BX                   ;set DX = port #
0752  B4 00                       MOV     AH,INIT_COM1            ;set AH = init function #
0754  B0 A3                       MOV     AL,0A3H                 ;init COM1 = 2400 baud,
0756  CD 14                       INT     COM1_TRAP               ;no parity, 1 stop bit and
0758  46                  NOCOM:  INC     SI                      ;incr pointers to next port
0759  46                          INC     SI                      ;
075A  47                          INC     DI                      ;
075B  47                          INC     DI                      ;
075C  43                          INC     BX                      ;incr port #
075D  EB DC                       JMP     SHORT INITCOMSLOOP      ;loop
075F  A1 0410 R           XITINITCOMS:  MOV     AX,EQUIP_FLAG     ;set equipment flag RS232 bits
0762  D0 E1                       SHL     CL,1                    ;(bits 11,10,9)
0764  80 E4 F1                    AND     AH,0F1H                 ;
0767  0A E1                       OR      AH,CL                   ;
0769  A3 0410 R                   MOV     EQUIP_FLAG,AX           ;
076C  BE 047C R                   MOV     SI,OFFSET RS232_TIMEOUT ;set timeout values for all
076F  B8 0101                     MOV     AX,0101H                ;
0772  89 04                       MOV     [SI],AX                 ;
0774  46                          INC     SI                      ;
0775  46                          INC     SI                      ;
0776  89 04                       MOV     [SI],AX                 ;
0778  C3                          RET                             ;
                          INITCOMS        ENDP                    ;

0779  03F8                TBL_COMPORT     DW      03F8H           ;port adr for com port 1
077B  02F8                                DW      02F8H           ;port adr for com port 2
077D  0000                                DW      0               ;end of table

                          ;------INITPRNS initializes any PC compatible PRINTER ports.

077F
077F  E8 03F1 R           INITPRNS        PROC    NEAR            ;
                                          CALL    DDS             ;set DS = IODATA
                                          ASSUME  DS:IODATA       ;
0782  BF 07C4 R                           MOV     DI,OFFSET TBL_PRNPORT   ;set SI = prnt port adr table
0785  BE 0408 R                           MOV     SI,OFFSET PRINTER_BASE  ;set DI = prntr port base
0788  B3 00                               MOV     BL,0            ;init printer cnt = 0
078A  2E: 8B 15           INITPRNSLOOP:   MOV     DX,CS:[DI]      ;get port adr from tbl
078D  83 FA 00                            CMP     DX,0            ;
0790  74 16                               JE      XITINITPRNS     ;if end of table, exit
0792  B0 AA                               MOV     AL,0AAH         ;
0794  EE                                  OUT     DX,AL           ;write pattern to port
0795  52                                  PUSH    DX              ;delay a bit
0796  5A                                  POP     DX              ;
0797  EC                                  IN      AL,DX           ;input port data
0798  52                                  PUSH    DX              ;wait a bit
0799  5A                                  POP     DX              ;
079A  3C AA                               CMP     AL,0AAH         ;
079C  75 04                               JNZ     NOPRN           ;if not correct, dont init
079E  FE C3                               INC     BL              ;inc # printers installed
07A0  89 14                               MOV     [SI],DX         ;store port adr
07A2  46                    NOPRN:        INC     SI              ;incr pointers to next port
07A3  46                                  INC     SI              ;
07A4  47                                  INC     DI              ;
07A5  47                                  INC     DI              ;
07A6  EB E2                               JMP     SHORT INITPRNSLOOP      ;loop
07A8  A1 0410 R           XITINITPRNS:    MOV     AX,EQUIP_FLAG   ;set equipment flag prntr bits
07AB  B1 06                               MOV     CL,6            ;(bits 15,14)
07AD  D2 E3                               SHL     BL,CL           ;
07AF  80 E4 3F                            AND     AH,3FH          ;
07B2  0A E3                               OR      AH,BL           ;
07B4  A3 0410 R                           MOV     EQUIP_FLAG,AX   ;
07B7  BE 0478 R                           MOV     SI,OFFSET PRNT_TIMEOUT  ;set timeout values for all
07BA  B8 1414                             MOV     AX,1414H        ;
07BD  89 04                               MOV     [SI],AX         ;
07BF  46                                  INC     SI              ;
```

```
07C0  46                        INC     SI                      ;
07C1  89 04                     MOV     [SI],AX                 ;
07C3  C3                        RET                             ;
                        INITPRNS        ENDP                    ;

07C4  03BC                      TBL_PRNPORT     DW      03BCH           ;port adr for prnt port 1
07C6  0378                                      DW      0378H           ;port adr for prnt port 2
07C8  0278                                      DW      0278H           ;port adr for prnt port 3
07CA  0000                                      DW      0               ;end of table

                        ;------SET_EQUIP will set game adapter if joystick found and
                        ;       set serial printer if no parallel printer installed.
07CC                    SET_EQUIP       PROC    NEAR            ;
07CC  E8 03F1 R                         CALL    DDS             ;set DS = iodata
                                        ASSUME  DS:IODATA       ;
07CF  8B 1E 0410 R                      MOV     BX,EQUIP_FLAG   ;get equip flag to BX
07D3  BA 0201                           MOV     DX,GAMEPORT     ;
07D6  EC                                IN      AL,DX           ;
07D7  A8 0F                             TEST    AL,0FH          ;see if joystick present
07D9  75 03                             JNZ     NOGAMES         ;if not, skip
07DB  80 CF 10                          OR      BH,10H          ;set game bit in flag
07DE  8B C3                     NOGAMES:        MOV     AX,BX           ;
07E0  80 E4 C0                          AND     AH,0C0H         ;see if any parallel printer
07E3  75 0A                             JNZ     XITEQUIP        ;if not, skip
07E5  A0 05D7 R                         MOV     AL,SCCFLAG      ;
07E8  0A C0                             OR      AL,AL           ;if no ser port, exit
07EA  74 03                             JZ      XITEQUIP        ;
07EC  80 CF 20                          OR      BH,20H          ;set serial printer bit
07EF  BA 0062                   XITEQUIP:       MOV     DX,COPRPORT
07F2  EC                                IN      AL,DX
07F3  3C 62                             CMP     AL,COPRID
07F5  74 04                             JZ      SETEQUIP
07F7  81 CB 0002                        OR      BX,0002H
07F8  89 1E 0410 R              SETEQUIP:       MOV     EQUIP_FLAG,BX   ;store equip flag
07FC  C3                                RET                     ;
                        SET_EQUIP       ENDP                    ;

                        ;*************************************************************************
                        ; INITRAMDISK                                                           *
                        ;               INITRAMDISK will find and format any ramdisks taht aren't *
                        ;               already formatted.                                      *
                        ;                                                                       *
                        ; Enter:       none                                                    *
                        ; EXIT:        none                                                    *
                        ; Alters:      AX,FL                                                   *
                        ; Calls:                                                               *
                        ; Stack:       3 words                                                 *
                        ;*************************************************************************
                                        ASSUME  CS:CODE, DS:IODATA, ES:NOTHING
0800                    INITRAMDISK     PROC    NEAR            ;
0800  06                                PUSH    ES              ;
0801  1E                                PUSH    DS              ;save register status
0802  57                                PUSH    DI              ;
0803  52                                PUSH    DX              ;
0804  51                                PUSH    CX              ;
0805  53                                PUSH    BX              ;
0806  BB 0001                           MOV     BX,0001H        ;init disk map = 0001
0809  33 C9                           XOR     CX,CX           ;clear the unit count
080B  B8 E000                           MOV     AX,DISKSEG      ;ES:DI points to disk memory
080E  8E C0                             MOV     ES,AX           ;   area
0810  BF 0000                           MOV     DI,0            ;
0813  BA 0280                           MOV     DX,ROMBASEPORT  ;sector cntrl port addr
0816  0E                                PUSH    CS              ;set DS = IOCODE
0817  1F                                POP     DS              ;
                                        ASSUME  DS:CODE         ;

0818  B0 00                    IDENTIFY:       MOV     AL,0            ;write sector 0 and card enable
081A  EE                                OUT     DX,AL           ;   to port
081B  B0 04                             MOV     AL,004H         ;
081D  42                                INC     DX              ;
081E  EE                                OUT     DX,AL           ;
081F  51                                PUSH    CX              ;
0820  8C C1                             MOV     CX,ES           ;
0822  E8 0432 R                         CALL    FIND_DISK       ;id in boot sector?
0825  75 08                             JNZ     SETDSKMASK      ;   -jump if so
0827  E8 03F9 R                         CALL    CHECK_RAM       ;no id - is it RAM?
```

```
082A  75 06                     JNZ     ID1                     ;   -jump if not
082C  E8 0861 R                 CALL    FORMAT_RAM              ;RAM w/no id - format it
082F  E8 0857 R         SETDSKMASK:     CALL    DISKMASK        ;or mask into flag
0832  81 FA 0283        ID1:    CMP     DX,ROMBASEPORT+3        ;
0836  75 0A                     JNE     NOTRAMDSKB              ;if not mem disk B, continue
0838  1E                        PUSH    DS                      ;
0839  E8 03F1 R                 CALL    DDS                     ;
                                ASSUME  DS:IODATA               ;set DS = iodata
083C  C6 06 06A2 R 01           MOV     RAMDRVBFLG,1            ;set ramdrvb flag
0841  1F                        POP     DS                      ;
                                ASSUME  DS:CODE                 ;
0842  59                NOTRAMDSKB:    POP     CX              ;
0843  41                       INC     CX                      ;increment disk count
0844  B0 00                    MOV     AL,0                    ;disable card
0846  EE                        OUT     DX,AL                   ;
0847  D1 E3                    SHL     BX,1                    ;rotate msak
0849  42                       INC     DX                      ;point to next card
084A  81 FA 02A0                CMP     DX,ROMBASEPORT+32       ;finished w/all cards?
084E  72 C8                     JC      IDENTIFY                ;check next card
0850  5B                        POP     BX                      ;
0851  59                        POP     CX                      ;
0852  5A                        POP     DX                      ;
0853  5F                        POP     DI                      ;
0854  1F                        POP     DS                      ;
0855  07                        POP     ES                      ;
0856  C3                        RET                             ;

0857  1E                DISKMASK:      PUSH    DS              ;or BX into ramdiskmask flag
0858  E8 03F1 R                 CALL    DDS                     ;
                                ASSUME  DS:IODATA               ;
085B  09 1E 06A3 R              OR      MEMDSKMASK,BX           ;
085F  1F                        POP     DS                      ;
0860  C3                        RET                             ;

                        INITRAMDISK     ENDP

                        ;*****************************************************************************
                        ; FORMAT_RAM                                                                 *
                        ;               Initializes ("formats") a RAM memory disk                   *
                        ;                                                                           *
                        ; Entry:       ES = memory disk segment (disk is enabled)                    *
                        ;               DS = IOCODE                                                   *
                        ;               DX = port address                                           *
                        ; Exit:        CO if format successful                                       *
                        ; Alters:      Flags                                                         *
                        ; Calls:
                        ; Stack:
                        ;*****************************************************************************
                                ASSUME  CS:CODE,DS:CODE,ES:NOTHING
0861                    FORMAT_RAM      PROC    NEAR            ;
0861  53                        PUSH    BX                      ;
0862  51                        PUSH    CX                      ;save unit count
0863  52                        PUSH    DX                      ;save port address

                            ;-----  zero all but boot and 1st FAT sectors:
0864  B9 0000                    MOV     CX,0                    ;start with sector 0
0867  81 EA 0280                SUB     DX,ROMBASEPORT          ;set DL = drive (card) number
086B  D1 CA                     ROR     DX,1                    ;
086D  B6 00                     MOV     DH,0                    ;
086F  1E                        PUSH    DS                      ;
0870  07                        POP     ES                      ;
0871  8D 1E 0A2C R      FR2:    LEA     BX,BLANK                ;ES:BX = xfer address
0875  B8 0301                   MOV     AX,0301H                ;write one sector of zeros
0878  CD 77                     INT     MDISK_TRAP              ;
087A  72 14                     JC      FR1                     ;jump when write fails
087C  41                        INC     CX                      ;
087D  80 FA 01                  CMP     DL,1
0880  74 08                     JE      CHK128K
0882  81 F9 0201                CMP     CX,513                  ;max RAM sectors is 512
0886  72 E9                     JB      FR2                     ;
0888  EB 06                     JMP     SHORT FR1
088A  81 F9 0101        CHK128K:        CMP     CX,257
088E  72 E1                     JB      FR2

                            ;----- write boot sector and 1st sector of FAT:
0890  49                FR1:    DEC     CX                      ;
```

```
0891  51                        PUSH    CX                      ;save # good sectors
0892  B8 0301                   MOV     AX,0301H                ;
0895  B9 0000                   MOV     CX,0                    ;
0898  8D 1E 0908 R              LEA     BX,RAMBOOTSECT          ;
089C  CD 77                     INT     MDISK_TRAP              ;
089E  72 18                     JC      FR5                     ;jump if error
08A0  B9 0001                   MOV     CX,1                    ;
08A3  B8 0301                   MOV     AX,0301H                ;
08A6  8D 1E 0A2C R              LEA     BX,BLANK                ;
08AA  CD 77                     INT     MDISK_TRAP              ;
08AC  72 0A                     JC      FR5                     ;jump if error

                        ;----- finished formatting:
08AE  B8 E000                   MOV     AX,DISKSEG              ;restore ES
08B1  8E C0                     MOV     ES,AX                   ;
08B3  F8                        CLC                             ;
08B4  EB 02 90                  JMP     FR5                     ;
08B7  F9                FR4:    STC                             ;carry indicates failure
08B8  59                FR5:    POP     CX                      ;restore # sectors
08B9  5A                        POP     DX                      ;
08BA  E8 08C0 R                 CALL    CHANGEBOOT
08BD  59                        POP     CX                      ;
08BE  5B                        POP     BX
08BF  C3                        RET                             ;
                        FORMAT_RAM      ENDP                    ;

                        ;------CHANGEBOOT will set RAMTOTALSECT, RAMFATSECT and RAMDIRENTRY in boot
                        ;       sector to valuse derived from CX (# good sectors).
                        ;       INPUT: CX = total # of good sectors
                        ;              DX = port address
08C0                    CHANGEBOOT      PROC    NEAR            ;
08C0  52                        PUSH    DX                      ;
08C1  53                        PUSH    BX                      ;
08C2  81 FA 0283                CMP     DX,ROMBASEPORT+3
08C6  B0 04                     MOV     AL,4                    ;select sector 0 of cur drive
08C8  75 02                     JNE     CHNGBOOT
08CA  0C 08                     OR      AL,08H
08CC  EE                CHNGBOOT:       OUT     DX,AL           ;
08CD  4A                        DEC     DX                      ;
08CE  B0 00                     MOV     AL,0                    ;
08D0  EE                        OUT     DX,AL                   ;
08D1  BB 0000                   MOV     BX,0                    ;
08D4  26: 89 8F 0013            MOV     ES:[BX].BL_BPB.TOTALSECTORS,CX ;store CX to # sectors
08D9  B8 0001                   MOV     AX,1                    ;
08DC  81 F9 0156                CMP     CX,342                  ;
08E0  72 01                     JC      CB1                     ;
08E2  40                        INC     AX                      ;
08E3  26: 89 87 0016    CB1:    MOV     ES:[BX].BL_BPB.SECTORSPERFAT,AX ;store # FAT sectors
08E8  B0 01                     MOV     AL,1                    ;enable 2nd sector (FAT)
08EA  EE                        OUT     DX,AL                   ;
08EB  BB 0000                   MOV     BX,0                    ;
08EE  B0 FE                     MOV     AL,0FEH                 ;store FE FF FF in FAT
08F0  26: 88 87 0000            MOV     ES:[BX].FATID1,AL       ;
08F5  FE C0                     INC     AL                      ;
08F7  26: 88 87 0001            MOV     ES:[BX].FATID2,AL       ;
08FC  26: 88 87 0002            MOV     ES:[BX].FATID3,AL       ;
0901  B0 00                     MOV     AL,0                    ;
0903  42                        INC     DX                      ;
0904  EE                        OUT     DX,AL                   ;disable card
0905  5B                        POP     BX                      ;
0906  5A                        POP     DX                      ;
0907  C3                        RET                             ;
                        CHANGEBOOT      ENDP                    ;

                        ;*****************************************************************************
                        ; BOOT                                                                       *
                        ;               This is the boot sector for the new memory disks, which     *
                        ;               includes the bootstrap loader program.  It reads the disk   *
                        ;               directory, finds and loads IO.SYS and MSDOS.SYS into memory,*
                        ;               and jumps to the start of IO.SYS.  It is loaded to and      *
                        ;               executes at segment BOOT_CODE, offset 0.                    *
                        ;                                                                           *
                        ; Entry:       BP = system memory size in paragraphs                       *
                        ;               DL = drive number (0 to 7, the first memory drive found)    *
                        ; Exit:        BX = last RAM paragraph (BP-1)                              *
                        ;               CL = 1 (default drive = A)
```

```
                        ;               AX = base segment of IO.SYS (0070H)                     *
                        ; Alters:
                        ; Calls:
                        ; Stack:
                        ;*****************************************************************************
0000  0003[             BOOTLOADER      STRUC                           ;New boot sector format
        ??              BL_START        DB      3 DUP (?)               ;
                      ]

0003  0008[             BL_OEMNAME      DB      8 DUP (?)               ;
        ??
                      ]

000B  0013[             BL_BPB          DB      (SIZE BPB) DUP (?)      ;
        ??
                      ]

001E  0003[             BL_NOTUSED      DW      3 DUP (?)               ;
        ????
                      ]

0024  000B[             BL_IODIR        DB      11 DUP (?)              ;
        ??
                      ]

002F  000B[             BL_MSDIR        DB      11 DUP (?)              ;
        ??
                      ]

003A                    BOOTLOADER      ENDS                            ;

                        FATID           STRUC
0000  ??                FATID1          DB      ?
0001  ??                FATID2          DB      ?
0002  ??                FATID3          DB      ?
0003                    FATID           ENDS

                                        ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
0908                    BOOT            PROC    FAR                     ;
0908  EB 51             RAMBOOTSECT:    JMP     SHORT LOAD_OS           ;boot sector for RAM disk
090A  00                                DB      0                       ;

090B  50 2D 4C 20 44 49 53              DB      'P-L DISK'              ;Pro-Log disk ID
      4B
0913  0200              BYTSPERSECT     DW      512                     ;BPB: bytes per sector
0915  01                               DB      1                       ;     sectors per cluster
0916  0001              RAMRSVSECT      DW      1                       ;     reserved sectors
0918  01                               DB      1                       ;     number of FATs
0919  0040              RAMDIRENTRY     DW      64                      ;     default directory entries
091B  0200              RAMTOTALSECT    DW      512                     ;     default total sectors
091D  FE                                DB      0FEH                    ;     media byte
091E  0002              RAMFATSECT      DW      2                       ;     default sectors/FAT

0920  0000              SECTTRACK       DW      0                       ;(not used)
0922  0000              NOHEADS         DW      0                       ;(not used)
0924  0000              SPLRSVSECT      DW      0                       ;(not used)

0926  49 4F 20 20 20 20 20 IOSYS_NAME    DB      'IO      SYS'           ;file name
      20 53 59 53
0931  4D 53 44 4F 53 20 20 MSSYS_NAME    DB      'MSDOS   SYS'           ;file name
      20 53 59 53

093C  28 43 29 43 6F 70 79              DB      '(C)Copyright 1986 Pro-Log Corp.'  ;legal stuff
      72 69 67 68 74 20 31
      39 38 36 20 50 72 6F
      20 4C 6F 67 20 43 6F
      72 70 2E

095B  0E                LOAD_OS:        PUSH    CS                      ;set ES and DS = CS
095C  07                                POP     ES                      ;
095D  0E                                PUSH    CS                      ;
095E  1F                                POP     DS                      ;
                                        ASSUME  DS:CODE,ES:CODE         ;

                        ;----- read the first sector of the directory into CS:200
```

```
095F  BE 0000                   MOV     SI,0                    ;(executes at offset 0)
0962  8B 4C 16                  MOV     CX,[SI].BL_BPB.SECTORSPERFAT ;get directory starting sector
0965  03 4C 0E                  ADD     CX,[SI].BL_BPB.RESERVEDSECTORS ;
0968  B8 0201                   MOV     AX,0201H                ;read one sector
096B  BB 0200                   MOV     BX,0200H                ;   to CS:200
096E  CD 77                     INT     MDISK_TRAP              ;

                        ;----- ignore 1st directory entry if it's a volume label:
0970  8B FB                     MOV     DI,BX                   ;ES:[DI] = 1st dir entry
0972  8A 45 0B                  MOV     AL,[DI].FILEATTRIBUTE   ;get file attribute
0975  24 08                     AND     AL,08H                  ;mask all but volume bit
0977  74 03                     JZ      NO_VOLUME               ;
0979  83 C7 20                  ADD     DI,020H                 ;point to second dir entry

                        ;----- look for IO.SYS:
097C  8D 36 0926 R      NO_VOLUME:      LEA     SI,IOSYS_NAME           ;DS:SI = correct entry
0980  81 EE 0908 R              SUB     SI,OFFSET RAMBOOTSECT   ;
0984  57                        PUSH    DI                      ;
0985  B9 000B                   MOV     CX,11                   ;11 bytes in a filename
0988  F3/ A6                    REPE    CMPSB                   ;compare filenames
098A  5F                        POP     DI                      ;
098B  E3 02                     JCXZ    OK1                     ;jump if filenames compare
098D  EB 4F                     JMP     SHORT BE1               ;no IO.SYS - error

098F  52                OK1:            PUSH    DX                      ;get size of IO.SYS in sectors
0990  83 7D 1E 00               CMP     [DI].FILESIZEHIGH,0     ;error if filesize > 64K bytes
0994  75 48                     JNE     BE1                     ;
0996  8B 45 1C                  MOV     AX,[DI].FILESIZELOW     ;get filesize in bytes
0999  BE 0000                   MOV     SI,0                    ;
099C  03 44 0B                  ADD     AX,[SI].BL_BPB.BYTESPERSECTOR  ;round up - add (1 sector - 1) bytes
099F  48                        DEC     AX                      ;
09A0  BA 0000                   MOV     DX,0                    ;extend AX to 32 bits
09A3  F7 74 0B                  DIV     [SI].BL_BPB.BYTESPERSECTOR  ;AX = number of sectors
09A6  5A                        POP     DX                      ;
09A7  8B D8                     MOV     BX,AX                   ;save in BX

                        ;----- look for MSDOS.SYS:
09A9  83 C7 20                  ADD     DI,20H                  ;next dir entry
09AC  8D 36 0931 R              LEA     SI,MSSYS_NAME           ;DS:SI = correct entry
09B0  81 EE 0908 R              SUB     SI,OFFSET RAMBOOTSECT   ;
09B4  57                        PUSH    DI                      ;
09B5  B9 000B                   MOV     CX,11                   ;11 bytes in a filename
09B8  F3/ A6                    REPE    CMPSB                   ;
09BA  5F                        POP     DI                      ;
09BB  E3 02                     JCXZ    OK2                     ;jump if filenames compare
09BD  EB 1F                     JMP     SHORT BE1               ;no MSDOS.SYS - error
09BF  52                OK2:            PUSH    DX                      ;get size of MSDOS.SYS in bytes
09C0  83 7D 1E 00               CMP     [DI].FILESIZEHIGH,0     ;error if filesize > 64K bytes
09C4  75 18                     JNE     BE1                     ;
09C6  8B 45 1C                  MOV     AX,[DI].FILESIZELOW     ;get filesize in bytes
09C9  BE 0000                   MOV     SI,0                    ;
09CC  03 44 0B                  ADD     AX,[SI].BL_BPB.BYTESPERSECTOR  ;round up - add (1 sector - 1) bytes
09CF  48                        DEC     AX                      ;
09D0  BA 0000                   MOV     DX,0                    ;extend AX to 32 bits
09D3  F7 74 0B                  DIV     [SI].BL_BPB.BYTESPERSECTOR  ;AX = number of sectors
09D6  5A                        POP     DX                      ;
09D7  03 C3                     ADD     AX,BX                   ;total sectors, IO.SYS+MSDOS.SYS
09D9  3D 0100                   CMP     AX,256                  ;can't be more than 255
09DC  72 02                     JB      OK3                     ;
09DE  EB 2E             BE1:            JMP     SHORT BE2               ;

                        ;----- read IO.SYS and MSDOS.SYS into memory at seg OS_BASE, offset 0:
09E0  53                OK3:            PUSH    BX                      ;save # sectors for IO.SYS
09E1  50                        PUSH    AX                      ;temp save total sectors (AL)
09E2  52                        PUSH    DX                      ;temp save drive number
09E3  B8 0020                   MOV     AX,32                   ;starting sector of IO.SYS:
09E6  BE 0000                   MOV     SI,0                    ;
09E9  F7 64 11                  MUL     [SI].BL_BPB.NUMBERDIRENTRIES  ;   (get bytes in directory)
09EC  03 44 0B                  ADD     AX,[SI].BL_BPB.BYTESPERSECTOR ;   (round up to next sector)
09EF  48                        DEC     AX                      ;
09F0  BA 0000                   MOV     DX,0                    ;       (sign extend for DIV)
09F3  F7 74 0B                  DIV     [SI].BL_BPB.BYTESPERSECTOR    ;   (get sectors/directory
09F6  8B C8                     MOV     CX,AX                   ;         ...to CX)
09F8  03 4C 16                  ADD     CX,[SI].BL_BPB.SECTORSPERFAT  ;   (plus FAT sectors)
09FB  03 4C 0E                  ADD     CX,[SI].BL_BPB.RESERVEDSECTORS  ; (plus reserved sectors)
09FE  5A                        POP     DX                      ;restore drive number
```

```
09FF  58                              POP     AX                      ;restore total  sectors (AL)
0A00  B4 02                           MOV     AH,2                    ;2 = read
0A02  BB ---- R                       MOV     BX,SEG OS_BASE          ;transfer address
0A05  8E C3                           MOV     ES,BX                   ;
0A07  BB 0000                         MOV     BX,0                    ;
0A0A  CD 77                           INT     MDISK_TRAP              ;
0A0C  73 02                           JNC     OK5                     ;
0A0E  EB 1B                   BE2:    JMP     SHORT BOOTERR           ;
0A10  5B                      OK5:    POP     BX                      ;restore # sectors for IO.SYS
0A11  73 02                           JNC     OK4                     ;
0A13  EB 16                           JMP     SHORT BOOTERR           ;

                              ;----- transfer control to IO.SYS:
0A15  8B C3                   OK4:    MOV     AX,BX                   ;size of IO.SYS in sectors
0A17  B1 05                           MOV     CL,5                    ;
0A19  D3 E0                           SHL     AX,CL                   ;size of IO.SYS in paragraphs
0A1B  05 ---- R                       ADD     AX,SEG OS_BASE          ;AX = base seg of MSDOS.SYS
0A1E  8B DD                           MOV     BX,BP                   ;BX = memory size, paragraphs
0A20  4B                              DEC     BX                      ;BX = last RAM paragraph
0A21  B1 08                           MOV     CL,8                    ;CL = default drive (H:)
0A23  83 C4 04                        ADD     SP,4                    ;adjust stack (no far return)
0A26  EA 0000 ---- R                  JMP     OS_BASE                 ;jump to IO.SYS

0A2B  CB                      BOOTERR:RET                             ;return to BIOS on error
                              BOOT    ENDP                            ;

                              ;*************************************************************************
                              ; This block of zeros is used in "formatting" new RAM disks.           *
                              ;*************************************************************************
0A2C  00                      BLANK   DB      0                       ;a sector of zeros
0A2D  00                      BLANK1  DB      0                       ;+++ THIS MAY CHANGE!!! +++
0A2E  00                      BLANK2  DB      0                       ;
0A2F  01FD[                           DB      509 DUP(0)              ;
        00
                ]



                              ;------TICK_INT (interupt 08h) increments cur tick count
                              ;             if 24 hour overflow, reset to 0 and set TIMER_OFLW flag

                                      ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
0C2C                          TICK_INT PROC   FAR                     ;
0C2C  1E                              PUSH    DS                      ;save registers
0C2D  50                              PUSH    AX                      ;
0C2E  B8 ---- R                       MOV     AX,SEG IODATA           ;point DS to RAM area
0C31  8E D8                           MOV     DS,AX                   ;
                                      ASSUME  DS:IODATA               ;
0C33  8C 16 069F R                    MOV     INTR_SS,SS              ;save current stack pointer
0C37  89 26 069D R                    MOV     INTR_SP,SP              ;     and segment
0C3B  8E D0                           MOV     SS,AX                   ;
0C3D  8D 26 0699 R                    LEA     SP,INTSTACK             ;
0C41  52                              PUSH    DX                      ;
0C42  FF 06 046C R                    INC     TIMER_LOW               ;incr ls word of tick count
0C46  75 04                           JNZ     NOLSOFLW                ;if no overflow...
0C48  FF 06 046E R                    INC     TIMER_HIGH              ;incr ms word of tick count
0C4C  83 3E 046E R 18         NOLSOFLW: CMP     TIMER_HIGH,MAXHOURSMS   ;see if max hours reached
0C51  75 18                           JNE     XIT_TICK_INT            ;if not...
0C53  81 3E 046C R 00B0               CMP     TIMER_LOW,MAXHOURSLS    ;
0C59  75 10                           JNE     XIT_TICK_INT            ;
0C5B  C7 06 046E R 0000               MOV     TIMER_HIGH,0            ;max time reached, reset tick
0C61  C7 06 046C R 0000               MOV     TIMER_LOW,0             ;count and set overflow flag
0C67  FF 06 0514 R                    INC     DAYSCOUNTED             ;incr # days counted
0C6B  80 3E 0440 R 00         XIT_TICK_INT: CMP MOTOR_COUNT,0           ;decr FDC motor timeout counter
0C70  74 06                           JZ      TURNDISKOFF             ;if not timeout, skip
0C72  FE 0E 0440 R                    DEC     MOTOR_COUNT             ;decr motot count
0C76  EB 0B                           JMP     SHORT NOTMOTTIME        ;
0C78  80 26 043F R F0         TURNDISKOFF: AND  MOTOR_STATUS,0F0H       ;turn off motor on bits in ststus
0C7D  B0 24                           MOV     AL,24H                  ;
0C7F  BA 02D8                         MOV     DX,PORTCNTL_FDC         ;
0C82  EE                              OUT     DX,AL                   ;
0C83  CD 1C                   NOTMOTTIME: INT   USERINT                 ;xfer control to user routine
0C85  B0 20                           MOV     AL,INTEND               ;
0C87  E6 20                           OUT     PORTCNTL_8259,AL        ;output end of interupt code
0C89  5A                              POP     DX                      ;
0C8A  8E 16 069F R                    MOV     SS,INTR_SS              ;restore stack
```

```
0C8E  8B 26 069D R            MOV     SP,INTR_SP              ;
0C92  58                      POP     AX                      ;
0C93  1F                      POP     DS                      ;
0C94  CF                      IRET                            ;
                      TICK_INT        ENDP                    ;

                      ;*** INT 14 ********************************************************
                      ;* COM1_INT                                                        *
                      ;*                                                                 *
                      ;*            These routine are for the 8250 com port onboard the 7888 card.*
                      ;*            INPUT: AH = 0 (init COM1)                            *
                      ;*                   Inputs DX = port number                       *
                      ;*                   Inputs AL = init code                         *
                      ;*                      Bits 7,6 & 5 = baud rate code              *
                      ;*                         000 = 110                               *
                      ;*                         001 = 150                               *
                      ;*                         010 = 300                               *
                      ;*                         011 = 600                               *
                      ;*                         100 = 1200                              *
                      ;*                         101 = 2400                              *
                      ;*                         110 = 4800                              *
                      ;*                         111 = 9600                              *
                      ;*                      Bits 4 & 3 = parity code                   *
                      ;*                         00 = none                               *
                      ;*                         01 = odd parity                         *
                      ;*                         10 = none                               *
                      ;*                         11 = even parity                        *
                      ;*                      Bit 2 = stop bit code                      *
                      ;*                         0 = one stop bit                        *
                      ;*                         1 = 2 stop bits                         *
                      ;*                      Bits 1 & 0 = char size code                *
                      ;*                         00 = not used                           *
                      ;*                         01 = not used                           *
                      ;*                         10 = 7 bit                              *
                      ;*                         11 = 8 bit                              *
                      ;*                   Outputs AX = communications status            *
                      ;*            INPUT: AH = 1 (send character)                       *
                      ;*                   Inputs  DX = port number                      *
                      ;*                           AL = char                             *
                      ;*                   Outputs AH = error code (bit 7 set = error)   *
                      ;*            INPUT: AH = 2 (recv character)                       *
                      ;*                   Inputs  DX = port number                      *
                      ;*                   Outputs AL = character                        *
                      ;*                           AH = error code (bit 7 = error)       *
                      ;*            INPUT: AH = 3 (get port status)                      *
                      ;*                   Inputs  DX = port number                      *
                      ;*                   Outputs AH = line status                      *
                      ;*                      Bit 7 = Time out error                     *
                      ;*                      Bit 6 = xfer shift reg empty               *
                      ;*                      Bit 5 = xfer holding reg empty             *
                      ;*                      Bit 4 = break detect error                 *
                      ;*                      Bit 3 = framing error                      *
                      ;*                      Bit 2 = parity error                       *
                      ;*                      Bit 1 = overrun error                      *
                      ;*                      Bit 0 = data ready                         *
                      ;*                   Outputs AL = modem status                     *
                      ;*                      Bit 7 = recvd line sig detect              *
                      ;*                      Bit 6 = ring indicator                     *
                      ;*                      Bit 5 = DSR                                *
                      ;*                      Bit 4 = CTS                                *
                      ;*                      Bit 3 = delta recv line sig detect         *
                      ;*                      Bit 2 = trailing edge ring detector        *
                      ;*                      Bit 1 = delta DSR                          *
                      ;*                      Bit 0 = delta CTS                          *
                      ;*            INPUT: AH = 69 (send char out int port)              *
                      ;*********************************************************************
= 0600                DIV75           EQU     1536
= 0417                DIV110          EQU     1047

                                      ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
0C95                  COM1_INT        PROC    FAR                     ;
0C95  FB                              STI                             ;turn interupts on
0C96  1E                              PUSH    DS                      ;save registers
0C97  56                              PUSH    SI                      ;
0C98  52                              PUSH    DX                      ;
0C99  51                              PUSH    CX                      ;
0C9A  53                              PUSH    BX                      ;
```

```
0C9B  8B F2                     MOV     SI,DX                   ;set SI = uart # word offset
0C9D  D1 E6                     SHL     SI,1                    ;
0C9F  E8 03F1 R                 CALL    DDS                     ;set DS = iodata seg
                                ASSUME  DS:IODATA               ;
0CA2  8B 94 0400 R              MOV     DX,RS232_BASE[SI]       ;get base port address
0CA6  0B D2                     OR      DX,DX                   ;see if 0
0CA8  74 14                     JZ      XITCOM1ERR              ;if so, error
0CAA  80 FC 69                  CMP     AH,69H                  ;
0CAD  74 50                     JE      @@CONXMIT               ;if func # = 69...
0CAF  0A E4                     OR      AH,AH                   ;
0CB1  74 13                     JZ      COM1INIT                ;if AH = 0, init 8250
0CB3  FE CC                     DEC     AH                      ;
0CB5  74 7B                     JZ      COM1WRITE               ;if AH = 1, write char
0CB7  FE CC                     DEC     AH                      ;
0CB9  75 3A                     JNZ     COM1STAT                ;if AH = 2, read char
0CBB  E9 0D61 R                 JMP     COM1RECV                ;go report com port status

0CBE  B4 80             XITCOM1ERR:     MOV     AH,80H          ;set error ret code
0CC0  58                XITCOM1:        POP     BX              ;restore registers & return
0CC1  59                                POP     CX              ;
0CC2  5A                                POP     DX              ;
0CC3  5E                                POP     SI              ;
0CC4  1F                                POP     DS              ;
0CC5  CF                                IRET                    ;

                        ;------COM1INIT sets the 8250 chip to the baud rate, parity status,
                        ;       stop bits & char size as specified in AL.
0CC6  50                COM1INIT:       PUSH    AX              ;save al status
0CC7  8A E8                             MOV     CH,AL           ;
0CC9  83 C2 03                          ADD     DX,3            ;send 80 hex to line control
0CCC  B0 80                             MOV     AL,80H          ;
0CCE  EE                                OUT     DX,AL           ;
0CCF  B1 05                             MOV     CL,5            ;move rate to ls 3 bits
0CD1  D2 ED                             SHR     CH,CL           ;
0CD3  8A CD                             MOV     CL,CH           ;
0CD5  B8 0417                           MOV     AX,DIV110       ;set divisor = 110 baud
0CD8  0A C9                             OR      CL,CL           ;
0CDA  74 05                             JZ      SETCOM1RATE     ;continue if 110 baud
0CDC  B8 0600                           MOV     AX,DIV75        ;set divisor = 75 baud
0CDF  D3 E8                             SHR     AX,CL           ;
0CE1  83 EA 03          SETCOM1RATE:    SUB     DX,3            ;output to divisor reg
0CE4  EE                                OUT     DX,AL           ;
0CE5  42                                INC     DX              ;
0CE6  8A C4                             MOV     AL,AH           ;
0CE8  EE                                OUT     DX,AL           ;
0CE9  58                                POP     AX              ;retrieve parameters
0CEA  24 1F                             AND     AL,1FH          ;strip off rate code
0CEC  42                                INC     DX              ;
0CED  42                                INC     DX              ;
0CEE  EE                                OUT     DX,AL           ;set length, parity & stop bits
0CEF  4A                                DEC     DX              ;
0CF0  4A                                DEC     DX              ;
0CF1  B0 00                             MOV     AL,0            ;
0CF3  EE                                OUT     DX,AL           ;disable interupts
0CF4  4A                                DEC     DX              ;point to base adr

                        ;------COM1STAT returns the line & modem status in AH,AL
0CF5  83 C2 05          COM1STAT:       ADD     DX,5            ;
0CF8  EC                                IN      AL,DX           ;get line & modem status
0CF9  8A E0                             MOV     AH,AL           ;AH = line status
0CFB  42                                INC     DX              ;
0CFC  EC                                IN      AL,DX           ;AL = modem status
0CFD  E8 C1                             JMP     XITCOM1         ;exit

                        ;------@@CONXMIT sends a char to int driven 8250 chip
0CFF  FA                @@CONXMIT:      cli
0D00  8A D8                             MOV     BL,AL           ;save char
0D02  83 C2 04                          ADD     DX;4            ;point to modem control
0D05  B0 0F                             MOV     AL,0FH          ;
0D07  EE                                OUT     DX,AL           ;set DTR, RTS, OUT1 & OUT2
0D08  42                                INC     DX              ;point to line status
0D09  42                                INC     DX              ;point to modem status
0D0A  EC                                IN      AL,DX           ;
0D0B  24 20                             AND     AL,20H          ;if DSR inactive, send it anyhow
0D0D  75 03                             JNZ     DSR_ACTIVE
0D0F  4A                                DEC     DX
```

```
0D10  EB 17                     JMP     SHORT @@SEND            ;
0D12  B7 30     DSR_ACTIVE:     MOV     BH,30H                  ;set BH = DSR & CTS
0D14  E8 0D82 R                 CALL    WAIT_READY              ;see if xmit ready
0D17  74 08                     JZ      @@CONDSR_RDY            ;if so...
0D19  8A C3     @@XMIT_TIME:    MOV     AL,BL                   ;restore AL with char
0D1B  80 CC 80                  OR      AH,80H                  ;return w/ time out error
0D1E  FB                        sti
0D1F  EB 9F                     JMP     XITCOM1                 ;

0D21  4A        @@CONDSR_RDY:   DEC     DX                      ;point to line status
0D22  B7 20                     MOV     BH,20H                  ;set BH = DTR
0D24  E8 0D82 R                 CALL    WAIT_READY              ;see if ready
0D27  75 F0                     JNZ     @@XMIT_TIME             ;if not, exit error
0D29  83 EA 05  @@SEND:         SUB     DX,5                    ;point to data
0D2C  8A C3                     MOV     AL,BL                   ;restore char to AL
0D2E  EE                        OUT     DX,AL                   ;send char & exit
0D2F  FB                        sti
0D30  EB 8E                     JMP     XITCOM1                 ;

                        ;------COM1WRITE sends a character to the 8250 uart chip
0D32  8A D8     COM1WRITE:      MOV     BL,AL                   ;save char
0D34  83 C2 04                  ADD     DX,4                    ;point to modem control
0D37  B0 03                     MOV     AL,3                    ;
0D39  EE                        OUT     DX,AL                   ;set DTR & RTS
0D3A  42                        INC     DX                      ;point to line status
0D3B  42                        INC     DX                      ;point to modem status
0D3C  EC                        IN      AL,DX                   ;
0D3D  24 20                     AND     AL,20H                  ;if DSR inactive, send it anyhow
0D3F  74 17                     JZ      COM1SEND                ;
0D41  B7 30                     MOV     BH,30H                  ;set BH = DSR & CTS
0D43  E8 0D82 R                 CALL    WAIT_READY              ;see if xmit ready
0D46  74 08                     JZ      DSR_READY               ;if so...
0D48  8A C3     XMIT_TIMEOUT:   MOV     AL,BL                   ;restore AL with char
0D4A  80 CC 80  COM1_TIMEOUT:   OR      AH,80H                  ;return w/ time out error
0D4D  E9 0CC0 R                 JMP     XITCOM1                 ;

0D50  4A        DSR_READY:      DEC     DX                      ;point to line status
0D51  B7 20                     MOV     BH,20H                  ;set BH = DTR
0D53  E8 0D82 R                 CALL    WAIT_READY              ;see if ready
0D56  75 F0                     JNZ     XMIT_TIMEOUT            ;if not, exit error
0D58  83 EA 05  COM1SEND:       SUB     DX,5                    ;point to data
0D5B  8A C3                     MOV     AL,BL                   ;restore char to AL
0D5D  EE                        OUT     DX,AL                   ;send char & exit
0D5E  E9 0CC0 R                 JMP     XITCOM1                 ;

                        ;------COM1RECV will receive a char from the 8250 to AL
0D61  83 C2 04  COM1RECV:       ADD     DX,4                    ;point to modem control
0D64  B0 03                     MOV     AL,3                    ;
0D66  EE                        OUT     DX,AL                   ;set DTR
0D67  42                        INC     DX                      ;point to modem status
0D68  42                        INC     DX                      ;
0D69  B7 20                     MOV     BH,20H                  ;set BH = DSR
0D6B  E8 0D82 R                 CALL    WAIT_READY              ;see if ready
0D6E  75 DA                     JNZ     COM1_TIMEOUT            ;if not ready, exit timeout
0D70  4A                        DEC     DX                      ;point to line status
0D71  B7 01                     MOV     BH,01H                  ;set BH = recv bfr full
0D73  E8 0D82 R                 CALL    WAIT_READY              ;see if ready
0D76  75 D2                     JNZ     COM1_TIMEOUT            ;if not, exit timeout
0D78  80 E4 1E                  AND     AH,1EH                  ;error conditione only
0D7B  83 EA 05                  SUB     DX,5                    ;point to data & get it to AL
0D7E  EC                        IN      AL,DX                   ;
0D7F  E9 0CC0 R                 JMP     XITCOM1                 ;exit

                        COM1_INT        ENDP

                        ;------WAIT_READY loops till the status line @DX of the 8250 matches
                        ;       the preset value in BH or till time out has occured.
0D82                    WAIT_READY      PROC    NEAR
0D82  53                                PUSH    BX                      ;save BX, for BL has char
0D83  D1 EE                             SHR     SI,1                    ;re-establish uart #
0D85  8A 9C 047C R                      MOV     BL,RS232_TIMEOUT[SI]    ;get outside loop count
0D89  33 C9                             XOR     CX,CX                   ;set inside loop count
0D8B  EC                WAIT_LOOP:      IN      AL,DX                   ;get & check status bits
0D8C  8A E0                             MOV     AH,AL                   ;
0D8E  22 C7                             AND     AL,BH                   ;
0D90  3A C7                             CMP     AL,BH                   ;
```

```
0D92  74 09                             JE      XITWAITRDY              ;if ready, exit
0D94  E2 F5                             LOOP    WAIT_LOOP               ;inside loop
0D96  FE CB                             DEC     BL                      ;
0D98  75 F1                             JNZ     WAIT_LOOP               ;outside loop
0D9A  80 CB 01                          OR      BL,1                    ;set NZ = time out
0D9D  5B                        XITWAITRDY:     POP     BX              ;
0D9E  C3                                RET                             ;
                                WAIT_READY      ENDP

                                ;*** INT 0C ********************************************************************
                                ;* COM1IN_INT                                                                 *
                                ;*         This service is called when a hardware interupt 0C occurs          *
                                ;*         which is the interupt generated when a char is received by         *
                                ;*         the CON when it has been redirected to the COM1 port.              *
                                ;*********************************************************************************
                                        ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
0D9F                            COM1IN_INT      PROC    FAR             ;
0D9F  1E                                        PUSH    DS              ;save registers
0DA0  50                                        PUSH    AX
0DA1  B8 ---- R                                 MOV     AX,SEG IODATA   ;point DS to RAM area
0DA4  8E D8                                     MOV     DS,AX           ;
                                                ASSUME  DS:IODATA       ;
0DA6  8C 16 069F R                              MOV     INTR_SS,SS      ;save current stack pointer
0DAA  89 26 069D R                              MOV     INTR_SP,SP      ;     and segment
0DAE  8E D0                                     MOV     SS,AX           ;
0DB0  8D 26 0699 R                              LEA     SP,INTSTACK     ;
0DB4  57                                        PUSH    DI              ;
0DB5  52                                        PUSH    DX              ;
0DB6  53                                        PUSH    BX              ;
0DB7  BA 03F8                                   MOV     DX,BASE_8250    ;set DX = 8250 base port adr
0DBA  83 C2 05                                  ADD     DX,5            ;point to line status reg
0DBD  EC                                        IN      AL,DX           ;
0DBE  8A E0                                     MOV     AH,AL           ;get line status to AH
0DC0  83 EA 05                                  SUB     DX,5            ;point back to data reg
0DC3  EC                                        IN      AL,DX           ;input data byte
0DC4  80 E4 1E                                  AND     AH,1EH          ;
0DC7  74 17                                     JZ      COM1INRDY       ;if no errors or break, store char
0DC9  80 E4 10                                  AND     AH,10H          ;
0DCC  75 4B                                     JNZ     COM1INBRK       ;if break...
0DCE  B0 20                             XITCOM1IN:      MOV     AL,INTEND       ;
0DD0  E6 20                                     OUT     PORTCNTL_8259,AL        ;output end of interupt code
0DD2  5B                                        POP     BX              ;
0DD3  5A                                        POP     DX              ;
0DD4  5F                                        POP     DI              ;
0DD5  8E 16 069F R                              MOV     SS,INTR_SS      ;restore stack
0DD9  8B 26 069D R                              MOV     SP,INTR_SP      ;
0DDD  58                                        POP     AX              ;restore registers
0DDE  1F                                        POP     DS              ;
0DDF  CF                                        IRET                    ;

0DE0  8B 1E 041C R                      COM1INRDY:      MOV     BX,KBD_BUF_TAIL         ;set BX = bufr tail
0DE4  8B FB                                     MOV     DI,BX           ;save it in DI
0DE6  3C 63                                     CMP     AL,"c"          ;see if ALT F5 (esc c) received
0DE8  75 14                                     JNE     NOCOM1_RESET    ;jump if not c
0DEA  4F                                        DEC     DI              ;
0DEB  8A 25                                     MOV     AH,[DI]         ;
0DED  47                                        INC     DI              ;
0DEE  80 FC 1B                                  CMP     AH,1BH          ;
0DF1  75 0B                                     JNE     NOCOM1_RESET    ;jump if prev char not esc
0DF3  8A 26 05D8 R                              MOV     AH,CON_RESET_ENABLE     ;
0DF7  0A E4                                     OR      AH,AH           ;
0DF9  74 03                                     JZ      NOCOM1_RESET    ;jump if reset not enabled
0DFB  E9 003C R                                 JMP     BIOS            ;reset system
0DFE  43                                NOCOM1_RESET:   INC     BX              ;inc tail & see if end
0DFF  81 FB 043E R                              CMP     BX,OFFSET KBD_BUF_END   ;
0E03  75 03                                     JNE     NOT_KBD_END     ;if not end...
0E05  BB 041E R                                 MOV     BX,OFFSET KBD_BUFFER    ;set BX = bufr start
0E08  3B 1E 041A R                      NOT_KBD_END:    CMP     BX,KBD_BUF_HEAD ;see if head = BX
0E0C  74 1D                                     JE      KBD_BUF_FULL    ;if so, bufr is full
0E0E  88 05                                     MOV     [DI],AL         ;store char in bufr at old tail
0E10  89 1E 041C R                              MOV     KBD_BUF_TAIL,BX ;store new tail
0E14  E8 0E31 R                                 CALL    CLR_CTS?        ;if bufr almost full, drop CTS
0E17  EB B5                                     JMP     SHORT XITCOM1IN ;exit

0E19  C7 06 041C R 041E R               COM1INBRK:      MOV     KBD_BUF_TAIL,OFFSET KBD_BUFFER;reset head & tail to bufr start
0E1F  C7 06 041A R 041E R                               MOV     KBD_BUF_HEAD,OFFSET KBD_BUFFER;
```

```
0E25  CD 1B                     INT     1BH                     ;go to user break interupt
0E27  B0 03                     MOV     AL,03H                  ;go put a ^C in bufr at 1st location
0E29  EB B5                     JMP     SHORT COM1INRDY         ;

0E2B  B0 07             KBD_BUF_FULL: MOV AL,BELL               ;
0E2D  CD 29                     INT     TTY_TRAP                ;send beep to console
0E2F  EB 9D                     JMP     SHORT XITCOM1IN         ;exit

                        COM1IN_INT      ENDP                    ;

                        ;------CLR_CTS? will see if less than 4 chars left in buffer. If so,
                        ;       CTS to 8250 will be dropped.
                                ASSUME  DS:IODATA               ;
0E31                    CLR_CTS?        PROC    NEAR            ;
0E31  8B 1E 041A R              MOV     BX,KBD_BUF_HEAD         ;set BX = # free bytes in bufr
0E35  2B 1E 041C R              SUB     BX,KBD_BUF_TAIL         ;
0E39  74 0E                     JZ      XITCLRCTS               ;
0E3B  73 03                     JNC     HEADMORE                ;
0E3D  83 C3 20                  ADD     BX,32                   ;
0E40  4B                HEADMORE:      DEC     BX              ;
0E41  83 FB 04                  CMP     BX,4                    ;
0E44  73 03                     JNC     XITCLRCTS               ;if # bytes more than 3, exit
0E46  E8 0E4A R                 CALL    CLR_CTS                 ;set CTS low
0E49  C3                XITCLRCTS:     RET                     ;
                        CLR_CTS?        ENDP

                                ASSUME  DS:IODATA               ;
0E4A                    CLR_CTS         PROC    NEAR            ;
0E4A  BA 03F8                   MOV     DX,BASE_8250            ;
0E4D  83 C2 04                  ADD     DX,4                    ;
0E50  EC                        IN      AL,DX                   ;
0E51  24 FD                     AND     AL,0FDH                 ;
0E53  EE                        OUT     DX,AL                   ;
0E54  C3                        RET                             ;
                        CLR_CTS         ENDP

                        ;*** INT 17 ********************************************************************
                        ;*                                                                              *
                        ;* PRINT_INT                                                                    *
                        ;*           This interupt service will interface to any PC compatible          *
                        ;*           printers installed.                                                *
                        ;* ENTRY:    DX = printer to be used                                            *
                        ;*           PRINTER_BASE data area set                                         *
                        ;*           AH = function to be performed                                      *
                        ;*                   AH = 00 = print char in AL                                 *
                        ;*                             returns status in AH                             *
                        ;*                   AH = 01 = init printer port                                *
                        ;*                             returns status in AH                             *
                        ;*                   AH = 02 = read printer status to AH                        *
                        ;*                             bit 7 = not busy                                 *
                        ;*                             bit 6 = acknowledge                              *
                        ;*                             bit 5 = out of paper                             *
                        ;*                             bit 4 = selected                                 *
                        ;*                             bit 3 = I/O error                                *
                        ;*                             bit 2,1 = not used                               *
                        ;*                             bit 0 = timeout                                  *
                        ;*                                                                              *
                        ;********************************************************************************

0E55                    PRINT_INT       PROC    FAR             ;
                                ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
0E55  FB                        STI                             ;turn interupts back on
0E56  1E                        PUSH    DS                      ;save registers
0E57  56                        PUSH    SI                      ;
0E58  52                        PUSH    DX                      ;
0E59  51                        PUSH    CX                      ;
0E5A  53                        PUSH    BX                      ;
0E5B  E8 03F1 R                 CALL    DDS                     ;set DS = IODATA
                                ASSUME  DS:IODATA               ;
0E5E  8B F2                     MOV     SI,DX                   ;set SI = printer #
0E60  8A 9C 0478 R              MOV     BL,PRNT_TIMEOUT[SI]     ;set BL = timeout value
0E64  D1 E6                     SHL     SI,1                    ;adj SI to printer base ptr
0E66  8B 94 0408 R              MOV     DX,PRINTER_BASE[SI]     ;set DX = printer base port
0E6A  83 FA 00                  CMP     DX,0                    ;
0E6D  74 12                     JE      WRPRNTERR               ;if port = 0, set error & exit
0E6F  0A E4                     OR      AH,AH                   ;
0E71  74 12                     JZ      PRINTOUT                ;if cmd = 0 (print char)...
0E73  FE CC                     DEC     AH                      ;
0E75  74 44                     JZ      PRINTINIT               ;if cmd = 1 (initialize)...
```

```
0E77  FE CC                     DEC     AH                      ;
0E79  74 35                     JZ      PRINTSTAT               ;if cmd = 2 (get status)...
                        XITPRINTINT:    POP     BX              ;restore registers & exit
0E7C  59                        POP     CX                      ;
0E7D  5A                        POP     DX                      ;
0E7E  5E                        POP     SI                      ;
0E7F  1F                        POP     DS                      ;
0E80  CF                        IRET                            ;

0E81  B4 01             WRPRNTERR:      MOV     AH,1            ;set err in AH & exit
0E83  EB F6                     JMP     SHORT XITPRINTINT
0E85  8A F8             PRINTOUT:       MOV     BH,AL           ;save char in BH
0E87  EE                        OUT     DX,AL                   ;output char to port
0E88  42                        INC     DX                      ;incr to status reg
0E89  33 C9             OUTSIDE:        XOR     CX,CX           ;set inside timeout loop count
0E8B  EC                INSIDE:         IN      AL,DX           ;input status to AH
0E8C  8A E0                     MOV     AH,AL                   ;
0E8E  A8 80                     TEST    AL,80H                  ;
0E90  75 13                     JNZ     STROBE                  ;if not busy...
0E92  E2 F7                     LOOP    INSIDE                  ;do inside delay loop
0E94  FE CB                     DEC     BL                      ;
0E96  75 F1                     JNZ     OUTSIDE                 ;do outside delay loop
0E98  80 CC 01                  OR      AH,1                    ;set timeout bit in AH
0E9B  80 E4 F9                  AND     AH,0F9H                 ;mask out unused bits
0E9E  8A C7             SETFLGXIT:      MOV     AL,BH           ;restore char to AL
0EA0  80 F4 48                  XOR     AH,48H                  ;some wierd bit swap for IBM
0EA3  EB D6                     JMP     SHORT XITPRINTINT       ;compatibility

0EA5  B0 0D             STROBE:         MOV     AL,0DH          ;set strobe bit high
0EA7  42                        INC     DX                      ;incr to reg C (strobe port)
0EA8  EE                        OUT     DX,AL                   ;output strobe high
0EA9  B0 0C             STROBEFINI:     MOV     AL,0CH          ;set strobe low
0EAB  EE                        OUT     DX,AL                   ;
0EAC  8A C7                     MOV     AL,BH                   ;restore char to AL
0EAE  4A                        DEC     DX                      ;set DX back to base port
0EAF  4A                        DEC     DX                      ;

0EB0  8A F8             PRINTSTAT:      MOV     BH,AL           ;save char in BH
0EB2  42                        INC     DX                      ;incr to status port
0EB3  EC                        IN      AL,DX                   ;input status to AH
0EB4  8A E0                     MOV     AH,AL                   ;
0EB6  80 E4 F8                  AND     AH,0F8H                 ;mask out unused bits & timeout
0EB9  EB E3                     JMP     SHORT SETFLGXIT         ;exit

0EBB  8A F8             PRINTINIT:      MOV     BH,AL           ;save char in BH
0EBD  42                        INC     DX                      ;incr to reg C port
0EBE  42                        INC     DX                      ;
0EBF  B0 08                     MOV     AL,8                    ;enable init function
0EC1  EE                        OUT     DX,AL                   ;
0EC2  B8 03E8                   MOV     AX,1000                 ;
0EC5  48                WAITINIT:       DEC     AX              ;delay a bit
0EC6  75 FD                     JNZ     WAITINIT                ;
0EC8  EB DF                     JMP     SHORT STROBEFINI        ;go get status

                        PRINT_INT       ENDP                    ;

;*** INT 1A ************************************************************************
;* CLOCK_INT: services for clock read & write                                        *
;*         INPUT: AH = 0 = read tick count                                           *
;*                 Returns tick count in CX:DX                                       *
;*                 Returns AL = 0 if no 24 hour overflow since last read             *
;*         INPUT: AH = 1 = set tick count                                            *
;*                 Inputs CX:DX = tick count                                         *
;*         INPUT: AH = 2 = READ TIME                                                 *
;*                 Returns: CH = hours (BCD)                                         *
;*                          CL = minutes (BCD)                                       *
;*                          DH = seconds (BCD)                                       *
;*                          DL = secs/100 (BCD)                                      *
;*         INPUT: AH = 3 = SET TIME                                                  *
;*                 Inputs:  CH = hours (BCD)                                         *
;*                          CL = minutes (BCD)                                       *
;*                          DH = seconds (BCD)                                       *
;*                          DL = 1 = daylight savings time                           *
;*         INPUT: AH = 4 = READ DATE                                                 *
;*                 Returns: CH = century (BCD)(19 or 20)                             *
;*                          CL = year (BCD)                                          *
;
```

```
                        ;*                      DH = month (BCD)                                *
                        ;*                      DL = day (BCD)                                  *
                        ;*      INPUT: AH = 5 = SET DATE                                        *
                        ;*                Inputs:  CH = century (BCD)(19 or 20)                 *
                        ;*                         CL = year (BCD)                              *
                        ;*                         DH = month (BCD)                             *
                        ;*                         DL = day (BCD)                               *
                        ;*************************************************************************

                                ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
0ECA                    CLOCK_INT       PROC    FAR                             ;
0ECA  1E                                PUSH    DS                              ;save DS
0ECB  50                                PUSH    AX
0ECC  E8 03F1 R                         CALL    DDS                             ;set DS = IODATA segment
                                        ASSUME  DS:IODATA                       ;
0ECF  80 FC 00                          CMP     AH,00H                          ;
0ED2  74 17                             JE      READ_TICK                       ;func 00, go read ticks
0ED4  FE CC                             DEC     AH                              ;
0ED6  74 26                             JZ      WRITE_TICK                      ;func 01, go write ticks
0ED8  FE CC                             DEC     AH                              ;
0EDA  74 32                             JZ      READ_TIME                       ;func 02, go read time
0EDC  FE CC                             DEC     AH                              ;
0EDE  74 44                             JZ      SET_TIME                        ;func 03, go set time
0EE0  FE CC                             DEC     AH                              ;
0EE2  74 48                             JZ      READ_DATE                       ;func 04, go read date
0EE4  FE CC                             DEC     AH                              ;
0EE6  74 4D                             JZ      SET_DATE                        ;func 05, go set date
0EE8  58                        XITCLOCKINT:    POP     AX                      ;
0EE9  1F                                POP     DS                              ;
0EEA  CF                                IRET                                    ;

                        ;------FUNCTION 0 (INT 1AH)
                        ;------Read current tick count from timer mailboxes to CX:DX
0EEB  FA                        READ_TICK:      CLI                             ;interupts off while reading
0EEC  A0 0470 R                                 MOV     AL,TIMER_OFLW           ;set AL = overflow flag status
0EEF  C6 06 0470 R 00                           MOV     TIMER_OFLW,0            ;reset overflow status
0EF4  8B 0E 046E R                              MOV     CX,TIMER_HIGH           ;set CX = hi word of tick count
0EF8  8B 16 046C R                              MOV     DX,TIMER_LOW            ;set DX = low word of tick count
0EFC  EB EA                                     JMP     SHORT XITCLOCKINT       ;exit

                        ;------FUNCTION 1 (INT 1AH)
                        ;------Write CX:DX to tick count of timer mailboxes
0EFE  FA                        WRITE_TICK:     CLI                             ;interupts off while writing
0EFF  C6 06 0470 R 00                           MOV     TIMER_OFLW,0            ;reset overflow status
0F04  89 0E 046E R                              MOV     TIMER_HIGH,CX           ;move CX to ticks high word
0F08  89 16 046C R                              MOV     TIMER_LOW,DX            ;move DX to ticks low word
0F0C  EB DA                                     JMP     SHORT XITCLOCKINT       ;exit

                        ;------FUNCTION 2 (INT 1A)
                        ;------Read time: Read tick count & convert to real time
0F0E  FA                        READ_TIME:      CLI
0F0F  8B 0E 046E R                              MOV     CX,TIMER_HIGH
0F13  8B 16 046C R                              MOV     DX,TIMER_LOW
0F17  C6 06 0470 R 00                           MOV     TIMER_OFLW,0
0F1C  E8 0F47 R                                 CALL    CNVRT_TICKS
0F1F  E8 10DE R                                 CALL    HEX_DWORD
0F22  EB C4                                     JMP     SHORT XITCLOCKINT

                        ;------FUNCTION 3 (INT 1A)
                        ;------Set Time: Convert time to tick count & store it
0F24  E8 0FF4 R                 SET_TIME:       CALL    BCDTOHEX
0F27  E8 0F92 R                                 CALL    CNVRT_TIME
0F2A  EB D2                                     JMP     SHORT WRITE_TICK

                        ;------FUNCTION 4 (INT 1A)
                        ;------Read Date: Read days-counted & convert to real date
0F2C  FA                        READ_DATE:      CLI
0F2D  A1 0514 R                                 MOV     AX,DAYSCOUNTED
0F30  E8 107A R                                 CALL    CONVERT_DAYS
0F33  EB B3                                     JMP     SHORT XITCLOCKINT

                        ;------FUNCTION 5 (INT 1A)
                        ;------Set Date: Convert real date to days since 1-1-80 & store
0F35  FA                        SET_DATE:       CLI
0F36  E8 102F R                                 CALL    CONVERT_DATE
0F39  A3 0514 R                                 MOV     DAYSCOUNTED,AX
```

```
0F3C  EB AA                     JMP     SHORT XITCLOCKINT

                        CLOCK_INT       ENDP

                        ;*****************************************************************************
                        ;* CNVRT_TICKS                                          7-23-86 dbc          *
                        ;*          CNVRT_TICKS will convert the tick count in CX:DX to real    *
                        ;*          time and return it to caller.                               *
                        ;* Enter:   CX:DX = tick count                                          *
                        ;* Exit:    CH = hours, CL = minutes, DH = seconds and DL = hundredths  *
                        ;* Alters:  FLAGS and CX:DX                                             *
                        ;* Calls:   none                                                        *
                        ;* Stack:   5 words                                                     *
                        ;*****************************************************************************

                        ;------The following constants are required for time conversions
0F3E  0005              FACTRFIVE       DW      5
0F40  E90B              TICKSPERHUN     DW      59659
0F42  0064              FACTRHUNS       DW      100
0F44  003C              FACTRSECS       DW      60
0F46  3C                FACTRMINS       DB      60

0F47                    CNVRT_TICKS     PROC    NEAR                    ;
0F47  50                                PUSH    AX                      ;save registers
0F48  53                                PUSH    BX                      ;
0F49  8B DA                             MOV     BX,DX                   ;move it to DX:AX and CX:BX
0F4B  8B C2                             MOV     AX,DX                   ;
0F4D  8B D1                             MOV     DX,CX                   ;
0F4F  03 C0                             ADD     AX,AX                   ;set DX:AX = cnt times 5
0F51  13 D2                             ADC     DX,DX                   ;
0F53  03 C0                             ADD     AX,AX                   ;
0F55  13 D2                             ADC     DX,DX                   ;
0F57  03 C3                             ADD     AX,BX                   ;
0F59  13 D1                             ADC     DX,CX                   ;
0F5B  2E: F7 36 0F40 R                  DIV     TICKSPERHUN             ;divide by ticks per hundred
0F60  8B C8                             MOV     CX,AX                   ;move hi word result to CX
0F62  33 C0                             XOR     AX,AX                   ;
0F64  2E: F7 36 0F40 R                  DIV     TICKSPERHUN             ;
0F69  91                                XCHG    AX,CX                   ;hundredths total is in AX:CX
0F6A  33 D2                             XOR     DX,DX                   ;
0F6C  2E: F7 36 0F42 R                  DIV     FACTRHUNS               ;
0F71  91                                XCHG    AX,CX                   ;hi result in CX
0F72  2E: F7 36 0F42 R                  DIV     FACTRHUNS               ;
0F77  8A DA                             MOV     BL,DL                   ;save hundredths results
0F79  8B D1                             MOV     DX,CX                   ;
0F7B  2E: F7 36 0F44 R                  DIV     FACTRSECS               ;
0F80  8A FA                             MOV     BH,DL                   ;calc & save seconds
0F82  2E: F6 36 0F46 R                  DIV     FACTRMINS               ;
0F87  8A CC                             MOV     CL,AH                   ;calc minutes to CL
0F89  8A E8                             MOV     CH,AL                   ;calc hours to CH
0F8B  8A F7                             MOV     DH,BH                   ;move seconds to DH
0F8D  8A D3                             MOV     DL,BL                   ;move hundredths to DL
0F8F  5B                                POP     BX                      ;
0F90  58                                POP     AX                      ;
0F91  C3                                RET                             ;
                        CNVRT_TICKS     ENDP                            ;
                        ;*****************************************************************************
                        ;* CNVRT_TIME                                           7-23-86 dbc          *
                        ;*          CNVRT_TIME will convert hours, minutes, seconds and         *
                        ;*          hundredths of seconds to a tickcount.                       *
                        ;* Enter:   CH=hours, CL=minutes, DH=seconds, and DL=hundredths         *
                        ;* Exit:    CX:DX = tick count                                          *
                        ;* Alters:  FLAGS                                                       *
                        ;* Calls:   none                                                        *
                        ;* Stack:   5 words                                                     *
                        ;*****************************************************************************

0F92                    CNVRT_TIME      PROC    NEAR                    ;
0F92  50                                PUSH    AX                      ;save registers
0F93  53                                PUSH    BX                      ;
0F94  8B DA                             MOV     BX,DX                   ;save secs & huns in BX
0F96  8A C5                             MOV     AL,CH                   ;
0F98  2E: F6 26 0F46 R                  MUL     FACTRMINS               ;mult hours by 60
0F9D  02 C1                             ADD     AL,CL                   ;add # minutes
0F9F  80 D4 00                          ADC     AH,0                    ;
0FA2  2E: F7 26 0F44 R                  MUL     FACTRSECS               ;mult seconds by 60
```

```
0FA7  02 C7                     ADD     AL,BH                   ;add # seconds
0FA9  80 D4 00                  ADC     AH,0                    ;
0FAC  83 D2 00                  ADC     DX,0                    ;
0FAF  8B CA                     MOV     CX,DX                   ;
0FB1  2E: F7 26 0F42 R          MUL     FACTRHUNS               ;mult by hundredths factor
0FB6  91                        XCHG    AX,CX                   ;
0FB7  53                        PUSH    BX                      ;push secs & huns on stack
0FB8  8B DA                     MOV     BX,DX                   ;
0FBA  2E: F7 26 0F42 R          MUL     FACTRHUNS               ;
0FBF  03 D8                     ADD     BX,AX                   ;
0FC1  58                        POP     AX                      ;pop secs & huns
0FC2  02 C8                     ADD     CL,AL                   ;add # hundredths
0FC4  80 D5 00                  ADC     CH,0                    ;
0FC7  83 D3 00                  ADC     BX,0                    ;
0FCA  8B C1                     MOV     AX,CX                   ;
0FCC  2E: F7 26 0F40 R          MUL     TICKSPERHUN             ;calculate tick count
0FD1  93                        XCHG    AX,BX                   ;
0FD2  8B CA                     MOV     CX,DX                   ;
0FD4  2E: F7 26 0F40 R          MUL     TICKSPERHUN             ;
0FD9  03 C1                     ADD     AX,CX                   ;
0FDB  83 D2 00                  ADC     DX,0                    ;
0FDE  8B C8                     MOV     CX,AX                   ;
0FE0  8B C2                     MOV     AX,DX                   ;
0FE2  33 D2                     XOR     DX,DX                   ;
0FE4  2E: F7 36 0F3E R          DIV     FACTRFIVE               ;
0FE9  91                        XCHG    CX,AX                   ;
0FEA  2E: F7 36 0F3E R          DIV     FACTRFIVE               ;
0FEF  8B D0                     MOV     DX,AX                   ;
0FF1  58                        POP     BX                      ;
0FF2  58                        POP     AX                      ;
0FF3  C3                        RET                             ;
                        CNVRT_TIME      ENDP                    ;

                        ;------This routine will convert BCD in CX & DX to binary and return
                        ;       values in same registers
0FF4                    BCDTOHEX        PROC    NEAR            ;
0FF4  8A C5                     MOV     AL,CH                   ;
0FF6  E8 1011 R                 CALL    BCDBYTE                 ;
0FF9  8A E8                     MOV     CH,AL                   ;
0FFB  8A C1                     MOV     AL,CL                   ;
0FFD  E8 1011 R                 CALL    BCDBYTE                 ;
1000  8A C8                     MOV     CL,AL                   ;
1002  8A C6                     MOV     AL,DH                   ;
1004  E8 1011 R                 CALL    BCDBYTE                 ;
1007  8A F0                     MOV     DH,AL                   ;
1009  8A C2                     MOV     AL,DL                   ;
100B  E8 1011 R                 CALL    BCDBYTE                 ;
100E  8A D0                     MOV     DL,AL                   ;
1010  C3                        RET                             ;
                        BCDTOHEX        ENDP                    ;

                        ;------This routine converts BCD in AL to binary in AL
1011                    BCDBYTE         PROC    NEAR            ;
1011  8A E0                     MOV     AH,AL                   ;
1013  80 E4 F0                  AND     AH,0F0H                 ;
1016  D0 CC                     ROR     AH,1                    ;
1018  D0 CC                     ROR     AH,1                    ;
101A  D0 CC                     ROR     AH,1                    ;
101C  D0 CC                     ROR     AH,1                    ;
101E  24 0F                     AND     AL,0FH                  ;
1020  D5 0A                     AAD                             ;
1022  C3                        RET                             ;
                        BCDBYTE         ENDP                    ;

                        ;------CONVERT_DATE will convert century (CH), year (CL), month (DH), and
                        ;       day to a count of # days since 1-1-1980

                        ;------Days-per-month table
1023  1F                DAYSPERMONTH    DB      31              ;days in JAN
1024  1C                LEAPMONTH       DB      28              ;       FEB
1025  1F                                DB      31              ;       MAR
1026  1E                                DB      30              ;       APR
1027  1F                                DB      31              ;       MAY
1028  1E                                DB      30              ;       JUN
1029  1F                                DB      31              ;       JUL
102A  1F                                DB      31              ;       AUG
```

```
102B  1E                                DB      30                      ;       SEP
102C  1F                                DB      31                      ;       OCT
102D  1E                                DB      30                      ;       NOV
102E  1F                                DB      31                      ;       DEC
= 1C89                          CENTURY_DAYS    EQU     7305            ;number days 1-1-80 to 12-31-99

102F                            CONVERT_DATE    PROC    NEAR            ;
102F  1E                                PUSH    DS                      ;save registers
1030  52                                PUSH    DX                      ;
1031  51                                PUSH    CX                      ;
1032  53                                PUSH    BX                      ;
1033  E8 0FF4 R                         CALL    BCDTOHEX                ;
1036  8C C8                            MOV     AX,CS                   ;set DS = CS
1038  8E D8                            MOV     DS,AX                   ;
103A  B8 0000                          MOV     AX,0                    ;init cnt = 0
103D  80 ED 13                         SUB     CH,19                   ;sub 1980 from CX
1040  80 E9 50                         SUB     CL,80                   ;
1043  80 FD 00                         CMP     CH,0                    ;
1046  74 03                            JE      CNVRT_YEAR              ;
1048  05 1C89                          ADD     AX,CENTURY_DAYS         ;
104B  80 F9 00                  CNVRT_YEAR:     CMP     CL,0            ;
104E  74 0D                            JZ      CNVRT_MONTH             ;if zero result, go
1050  05 016D                          ADD     AX,365                  ;add 365 to count
1053  E8 10CE R                         CALL    CHK_LEAP                ;
1056  75 01                            JNZ     NOT_LEAP                ;
1058  40                                INC     AX                      ;
1059  FE C9                   NOT_LEAP:       DEC     CL              ;if leap year, add 1 more
105B  EB EE                            JMP     SHORT CNVRT_YEAR        ;
105D  8D 1E 1023 R              CNVRT_MONTH:    LEA     BX,DAYSPERMONTH ;init ptr = days per month tbl
1061  80 FE 01                  CVT_MONTH:      CMP     DH,1            ;when month = jan, go
1064  74 0B                            JE      CNVRT_DAYS              ;
1066  8A 0F                            MOV     CL,[BX]                 ;add tbl value to count
1068  B5 00                            MOV     CH,0                    ;
106A  03 C1                            ADD     AX,CX                   ;
106C  43                                INC     BX                      ;
106D  FE CE                            DEC     DH                      ;
106F  EB F0                            JMP     SHORT CVT_MONTH         ;
1071  B6 00                     CNVRT_DAYS:    MOV     DH,0            ;
1073  03 C2                            ADD     AX,DX                   ;add day of month to cnt
1075  5B                                POP     BX                      ;
1076  59                                POP     CX                      ;
1077  5A                                POP     DX                      ;
1078  1F                                POP     DS                      ;
1079  C3                                RET                             ;

                                CONVERT_DATE    ENDP

                                ;*****************************************************************************
                                ;* CONVERT_DAYS                                                              *
                                ;*              This routine will convert # days in AX (since 1-1-1980)      *
                                ;*              to hex date.                                                 *
                                ;* ENTRY:       AX = days count since 1-1-1980                               *
                                ;* EXIT:        CH = century                                                 *
                                ;*              CL = year                                                    *
                                ;*              DH = month                                                   *
                                ;*              DL = day                                                     *
                                ;*****************************************************************************
107A                            CONVERT_DAYS    PROC    NEAR
                                                ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
107A  56                                PUSH    SI                      ;
107B  53                                PUSH    BX                      ;
107C  40                                INC     AX                      ;
107D  3D 1C89                           CMP     AX,CENTURY_DAYS         ;
1080  72 09                             JC      THISCENTURY             ;if in this century, go
1082  B5 14                            MOV     CH,20                   ;set century = 20
1084  B1 00                            MOV     CL,0                    ;set year = 0
1086  2D 1C89                          SUB     AX,CENTURY_DAYS         ;subtract century # days
1089  EB 04                            JMP     SHORT YEARLOOP          ;
108B  B5 13                     THISCENTURY:    MOV     CH,19           ;set century = 19
108D  B1 50                            MOV     CL,80                   ;set year = 80
108F  BB 016D                   YEARLOOP:       MOV     BX,365          ;set sub value = 365
1092  E8 10CE R                         CALL    CHK_LEAP                ;see if leap year
1095  75 01                            JNZ     NOTLEAPYEAR             ;if not, skip
1097  43                                INC     BX                      ;incr sub value = 366
1098  3B C3                   NOTLEAPYEAR:    CMP     AX,BX           ;see if day count < sub value
109A  72 06                             JC      SETMONTH                ;if so, year is done
```

```
109C  2B C3                     SUB     AX,BX                   ;sub a years worth from count
109E  FE C1                     INC     CL                      ;incr year
10A0  EB ED                     JMP     SHORT YEARLOOP          ;loop
10A2  8D 36 1023 R      SETMONTH:       LEA     SI,DAYSPERMONTH         ;set SI ptr = days/month tbl
10A6  B6 01                     MOV     DH,1                    ;set month = 1
10A8  2E: 8A 1C         MONTHLOOP:      MOV     BL,CS:[SI]              ;get # days/cur month
10AB  B7 00                     MOV     BH,0                    ;
10AD  3B C3                     CMP     AX,BX                   ;see if cnt < days in month
10AF  72 15                     JC      SETDAYS                 ;if so... month done
10B1  2B C3                     SUB     AX,BX                   ;sub # days from count
10B3  8D 1E 1024 R              LEA     BX,LEAPMONTH            ;see if ptr = february
10B7  3B DE                     CMP     BX,SI                   ;
10B9  75 06                     JNE     NOTLEAPMONTH            ;if not...
10BB  E8 10CE R                 CALL    CHK_LEAP                ;see if year is leapyear
10BE  75 01                     JNZ     NOTLEAPMONTH            ;if not...
10C0  48                        DEC     AX                      ;sub one more from count
10C1  46                NOTLEAPMONTH:   INC     SI              ;incr pointer
10C2  FE C6                     INC     DH                      ;incr month
10C4  EB E2                     JMP     SHORT MONTHLOOP         ;loop
10C6  8A D0             SETDAYS:        MOV     DL,AL           ;set days = whatever is left
10C8  5B                        POP     BX                      ;
10C9  5E                        POP     SI                      ;
10CA  E8 10DE R                 CALL    HEX_DWORD
10CD  C3                        RET                             ;
                        CONVERT_DAYS    ENDP                    ;

                        ;------CHK_LEAP returns ZR if leap year
                                ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
10CE                    CHK_LEAP        PROC    NEAR
10CE  51                        PUSH    CX                      ;
10CF  50                        PUSH    AX                      ;if CL is divisable by 4,
10D0  8A C1                     MOV     AL,CL                   ;
10D2  B4 00                     MOV     AH,0                    ;
10D4  B1 04                     MOV     CL,4                    ;
10D6  F6 F1                     DIV     CL                      ;
10D8  80 FC 00                  CMP     AH,0                    ;
10DB  58                        POP     AX                      ;
10DC  59                        POP     CX                      ;
10DD  C3                        RET                             ;
                        CHK_LEAP        ENDP

                        ;*****************************************************************************
                        ;*                                                                           *
                        ;* HEX_DWORD                                                                 *
                        ;*       This routine will convert the hex valuse in CX,DX to BCD            *
                        ;*                                                                           *
                        ;* ENTRY:  CX,DX = hex values                                                *
                        ;* EXIT:   CX,DX = BCD values                                                *
                        ;*****************************************************************************
                                ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
10DE                    HEX_DWORD       PROC    NEAR            ;
10DE  8A C5                     MOV     AL,CH                   ;
10E0  E8 10FB R                 CALL    HEXBYTE                 ;convert CH to BCD
10E3  8A E8                     MOV     CH,AL                   ;
10E5  8A C1                     MOV     AL,CL                   ;
10E7  E8 10FB R                 CALL    HEXBYTE                 ;convert CL to BCD
10EA  8A CB                     MOV     CL,AL                   ;
10EC  8A C6                     MOV     AL,DH                   ;
10EE  E8 10FB R                 CALL    HEXBYTE                 ;convert DH to BCD
10F1  8A F0                     MOV     DH,AL                   ;
10F3  8A C2                     MOV     AL,DL                   ;
10F5  E8 10FB R                 CALL    HEXBYTE                 ;convert DL to BCD
10F8  8A D0                     MOV     DL,AL                   ;
10FA  C3                        RET                             ;

10FB  53                HEXBYTE:        PUSH    BX              ;convert AL to BCD
10FC  BB 0000                   MOV     BX,0                    ;
10FF  3C 00             LOOP1:          CMP     AL,0            ;
1101  74 0F                     JE      XITHEXBYTE              ;
1103  FE C3                     INC     BL                      ;
1105  FE C8                     DEC     AL                      ;
1107  80 FB 0A                  CMP     BL,0AH                  ;
110A  75 F3                     JNE     LOOP1                   ;
110C  FE C7                     INC     BH                      ;
110E  B3 00                     MOV     BL,0                    ;
1110  EB ED                     JMP     SHORT LOOP1             ;
1112  D0 C7             XITHEXBYTE:     ROL     BH,1            ;
1114  D0 C7                     ROL     BH,1                    ;
```

```
 1116  D0 C7                          ROL     BH,1                    ;
 1118  D0 C7                          ROL     BH,1                    ;
 111A  80 E7 F0                       AND     BH,0F0H                 ;
 111D  0A DF                          OR      BL,BH                   ;
 111F  8A C3                          MOV     AL,BL                   ;
 1121  5B                             POP     BX                      ;
 1122  C3                             RET                             ;
                              HEX_DWORD       ENDP

                              ;*************************************************************************
                              ;* INIT2SCC                                                              *
                              ;*              Initializes data areas and interrupt vector table entries *
                              ;*              for 7314 or 7312 SCC ports A and B to be SER1 and SER2.  *
                              ;*              Also initializes interupt vectors for WRITE TTY and      *
                              ;*              SCC BIOS.                                                *
                              ;* Entry:       none                                                     *
                              ;* Exit:        none                                                     *
                              ;* Alters:      DS, DI, SI, BX, FLAGS                                    *
                              ;* Calls:       MOVEADDR                                                 *
                              ;* Stack:       03H words                                                *
                              ;*************************************************************************
                              ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
 1123                         INIT2SCC        PROC    NEAR                    ;
 1123  33 DB                                  XOR     BX,BX                   ;set DS = 0
 1125  8E DB                                  MOV     DS,BX                   ;
                                              ASSUME  DS:NOTHING              ;
                                                                              ;init int vect table for SER1
 1127  BF 0230 R                              MOV     DI,OFFSET R14A_TRAP     ;table entry for rcv char int, ch A
 112A  BB 1519 R                              MOV     BX,OFFSET SER1_INT      ;
 112D  E8 12C3 R                              CALL    MOVEADDR                ;
 1130  BF 0238 R                              MOV     DI,OFFSET S14A_TRAP     ;table entry for spcl rcv int, ch A
 1133  BB 1519 R                              MOV     BX,OFFSET SER1_INT      ;
 1136  E8 12C3 R                              CALL    MOVEADDR                ;
 1139  BF 0228 R                              MOV     DI,OFFSET E14A_TRAP     ;table entry for ext/stat int, ch A
 113C  BB 1691 R                              MOV     BX,OFFSET SER1_BRKINT   ;
 113F  E8 12C3 R                              CALL    MOVEADDR                ;
                                                                              ;init int vect table for SER2
 1142  BF 0210 R                              MOV     DI,OFFSET R14B_TRAP     ;table entry for rcv char int, ch B
 1145  BB 1541 R                              MOV     BX,OFFSET SER2_INT      ;
 1148  E8 12C3 R                              CALL    MOVEADDR                ;
 114B  BF 0218 R                              MOV     DI,OFFSET S14B_TRAP     ;table entry for spcl rcv int, ch B
 114E  BB 1541 R                              MOV     BX,OFFSET SER2_INT      ;
 1151  E8 12C3 R                              CALL    MOVEADDR                ;
 1154  BF 0208 R                              MOV     DI,OFFSET E14B_TRAP     ;table entry for ext/stat int, ch B
 1157  BB 1689 R                              MOV     BX,OFFSET SER2_BRKINT   ;
 115A  E8 12C3 R                              CALL    MOVEADDR                ;

 115D  BF 01C4 R                              MOV     DI,OFFSET VECTSCC_TRAP  ;tbl entry for common SCC bios routine
 1160  BB 12CB R                              MOV     BX,OFFSET SCCBIOS       ;
 1163  E8 12C3 R                              CALL    MOVEADDR                ;
 1166  BF 00A4 R                              MOV     DI,OFFSET TTY_VECT      ;table entry for INT 29H write TTY routine
 1169  BB 14B6 R                              MOV     BX,OFFSET WRITE_TTY     ;
 116C  E8 12C3 R                              CALL    MOVEADDR                ;

 116F  E8 03F1 R                              CALL    DDS                     ;get RAM data segment
                                              ASSUME  DS:IODATA               ;
                                                                              ;set up data area for SER1:
 1172  BE 0516 R                              MOV     SI,OFFSET SER1_DATA     ;point to data area
 1175  C7 04 00B1                            MOV     [SI].PORT_ADDRESS,CHA14 ;store port address
 1179  C6 44 0C 01                           MOV     [SI].TIMEOUT,1          ;set timeout to minimum
                                                                              ;set up data area for SER2:
 117D  BE 0525 R                              MOV     SI,OFFSET SER2_DATA     ;point to data area
 1180  C7 04 00B3                            MOV     [SI].PORT_ADDRESS,CHB14 ;store port address
 1184  C6 44 0C 01                           MOV     [SI].TIMEOUT,1          ;set timeout to minimum
 1188  C3                                    RET                             ;
                              INIT2SCC        ENDP                            ;

                              ;*************************************************************************
                              ;* INITMORSCC                                                            *
                              ;*              Initializes data areas and interrupt vector table entries *
                              ;*              for 7314 SCC ports C and D to be SER3 and SER4.          *
                              ;*                                                                       *
                              ;* Entry:       none                                                     *
                              ;* Exit:        none                                                     *
                              ;* Alters:      DS, DI, SI, BX, FLAGS                                    *
                              ;* Calls:       MOVEADDR                                                 *
```

```
                        ;* Stack:     03H words                                          *
                        ;*****************************************************************************
                                ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
1189                    INITMORSCC      PROC    NEAR                    ;
1189  33 DB                             XOR     BX,BX                   ;set DS = 0
118B  8E DB                             MOV     DS,BX                   ;
                                        ASSUME  DS:NOTHING              ;
                                                                        ;init int vect tbl for SER3
118D  BF 0234 R                         MOV     DI,OFFSET R14C_TRAP     ;tbl entry for rcv char int, ch C
1190  BB 1569 R                         MOV     BX,OFFSET SER3_INT      ;
1193  E8 12C3 R                         CALL    MOVEADDR                ;
1196  BF 023C R                         MOV     DI,OFFSET S14C_TRAP     ;tbl entry for spcl rcv int, ch C
1199  BB 1569 R                         MOV     BX,OFFSET SER3_INT      ;
119C  E8 12C3 R                         CALL    MOVEADDR                ;
119F  BF 022C R                         MOV     DI,OFFSET E14C_TRAP     ;tbl entry for ext/stat int, ch C
11A2  BB 16E1 R                         MOV     BX,OFFSET SER3_BRKINT   ;
11A5  E8 12C3 R                         CALL    MOVEADDR                ;
                                                                        ;init int vect tbl for SER4
11A8  BF 0214 R                         MOV     DI,OFFSET R14D_TRAP     ;tbl entry for rcv char int, ch D
11AB  BB 1591 R                         MOV     BX,OFFSET SER4_INT      ;
11AE  E8 12C3 R                         CALL    MOVEADDR                ;
11B1  BF 021C R                         MOV     DI,OFFSET S14D_TRAP     ;tbl entry for spcl rcv int, ch D
11B4  BB 1591 R                         MOV     BX,OFFSET SER4_INT      ;
11B7  E8 12C3 R                         CALL    MOVEADDR                ;
11BA  BF 020C R                         MOV     DI,OFFSET E14D_TRAP     ;tbl entry for ext/stat int, ch D
11BD  BB 1709 R                         MOV     BX,OFFSET SER4_BRKINT   ;
11C0  E8 12C3 R                         CALL    MOVEADDR                ;
                                                                        ;data area initialization:
11C3  E8 03F1 R                         CALL    DDS                     ;get RAM data segment
                                        ASSUME  DS:IODATA               ;
                                                                        ;set up data area for SER3
11C6  BE 0534 R                         MOV     SI,OFFSET SER3_DATA     ;point to data area
11C9  C7 04 00B5                        MOV     [SI].PORT_ADDRESS,CHC14 ;store port address
11CD  C6 44 0C 01                       MOV     [SI].TIMEOUT,1          ;set timeout to minimum
                                                                        ;set up data area for SER4
11D1  BE 0543 R                         MOV     SI,OFFSET SER4_DATA     ;point to data area
11D4  C7 04 00B7                        MOV     [SI].PORT_ADDRESS,CHD14 ;store port address
11D8  C6 44 0C 01                       MOV     [SI].TIMEOUT,1          ;set timeout to minimum
11DC  C3                                RET                             ;
                        INITMORSCC      ENDP                            ;

                        ;*****************************************************************************
                        ;* INITSCC                                                                   *
                        ;*          Initializes SCC devices on 7312 or 7314 according to             *
                        ;*          data areas previously initialized, by writing init table         *
                        ;*          strings to SCCs using INT SCC_TRAP                               *
                        ;*                                                                           *
                        ;* Entry:   Data areas and interrupt vector table entries initialized        *
                        ;* Exit:    none                                                             *
                        ;* Alters:  DS, BX, CX, SI, BP, AH, FLAGS                                    *
                        ;* Calls:   INT SCC_TRAP                                                     *
                        ;* Stack:   0DH words                                                        *
                        ;*****************************************************************************
                                ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
11DD                    INITSCC         PROC    NEAR                    ;
11DD  0E                                PUSH    CS                      ;set ES = CS
11DE  07                                POP     ES                      ;
11DF  E8 03F1 R                         CALL    DDS                     ;point to RAM data areas
                                        ASSUME  DS:IODATA,ES:CODE       ;SER1 SCC initialization
11E2  BD 0516 R                         MOV     BP,OFFSET SER1_DATA     ;get data area offset
11E5  3E: 8B 5E 00                      MOV     BX,DS:[BP].PORT_ADDRESS ;get port address
11E9  23 DB                             AND     BX,BX                   ;
11EB  74 51                             JZ      XITINISCC               ;if port adr = 0000, exit
11ED  BB 0000                           MOV     BX,SER1_DEV_NUM         ;load SCC device number
11F0  BF 1A26 R                         MOV     DI,OFFSET SER1TABLE     ;load offset of init table
11F3  2E: 8B 0D                         MOV     CX,CS:[DI]              ;number of bytes to CX
11F6  83 C7 02                          ADD     DI,2                    ;point DI to first data byte
11F9  B4 00                             MOV     AH,SCC_INIT             ;
11FB  CD 71                             INT     SCC_TRAP                ;
                                                                        ;SER2 SCC initialization:
11FD  BD 0525 R                         MOV     BP,OFFSET SER2_DATA     ;get data area offset
1200  BB 0001                           MOV     BX,SER2_DEV_NUM         ;load SCC device number
1203  BF 1A44 R                         MOV     DI,OFFSET SER2TABLE     ;load offset of init table
1206  2E: 8B 0D                         MOV     CX,CS:[DI]              ;number of bytes to CX
1209  83 C7 02                          ADD     DI,2                    ;point DI to first data byte
120C  B4 00                             MOV     AH,SCC_INIT             ;
```

```
120E  CD 71                             INT     SCC_TRAP                ;
                                                                        ;SER3 SCC initialization:
1210  BD 0534 R                         MOV     BP,OFFSET SER3_DATA     ;get data area offset
1213  3E: 8B 5E 00                      MOV     BX,DS:[BP].PORT_ADDRESS ;get port address
1217  23 DB                             AND     BX,BX                   ;
1219  74 23                             JZ      XITINISCC               ;if port adr = 0000, exit
121B  BB 0002                           MOV     BX,SER3_DEV_NUM         ;load SCC device number
121E  BF 1A62 R                         MOV     DI,OFFSET SER3TABLE     ;load offset of init table
1221  2E: 8B 0D                        MOV     CX,CS:[DI]              ;number of bytes to CX
1224  83 C7 02                         ADD     DI,2                    ;point DI to first data byte
1227  B4 00                            MOV     AH,SCC_INIT             ;
1229  CD 71                             INT     SCC_TRAP                ;
                                                                        ;SER4 SCC initialization:
122B  BD 0543 R                         MOV     BP,OFFSET SER4_DATA     ;get data area offset
122E  BB 0003                           MOV     BX,SER4_DEV_NUM         ;load SCC device number
1231  BF 1A62 R                         MOV     DI,OFFSET SER4TABLE     ;load offset of init table
1234  2E: 8B 0D                        MOV     CX,CS:[DI]              ;number of bytes to CX
1237  83 C7 02                         ADD     DI,2                    ;point DI to first data byte
123A  B4 00                            MOV     AH,SCC_INIT             ;
123C  CD 71                             INT     SCC_TRAP                ;
123E  C3                        XITINISCC:      RET                     ;return
                                INITSCC         ENDP                    ;

                                ;*************************************************************************
                                ;* BUFINIT                                                               *
                                ;*              Initializes pointers for all character device input ring  *
                                ;*              buffers; initializes handshake and timeout flags for each *
                                ;*              device                                                   *
                                ;*                                                                       *
                                ;* Entry:       none                                                     *
                                ;* Exit:        none                                                     *
                                ;* Alters:      DS, SI                                                   *
                                ;* Calls:       none                                                     *
                                ;* Stack:       03H words                                                *
                                ;*************************************************************************
                                        ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
123F                            BUFINIT PROC    NEAR                    ;init buffer pointers
123F  E8 03F1 R                         CALL    DDS                     ;get RAM data segment
                                        ASSUME  DS:IODATA               ;
                                                                        ;set up SER1 buffer:
1242  BE 0516 R                         MOV     SI,OFFSET SER1_DATA     ;point to data area
1245  C7 44 02 ---- R                   MOV     [SI].BUFFER_SEG,SEG IODATA ;
124A  C7 44 04 0552 R                   MOV     [SI].BUFFER_START,OFFSET BUF1 ;init buffer pointers
124F  C7 44 06 0572 R                   MOV     [SI].BUFFER_END,OFFSET BUF1_END;
1254  C7 44 08 0552 R                   MOV     [SI].BUFFER_HEAD,OFFSET BUF1 ;
1259  C7 44 0A 0552 R                   MOV     [SI].BUFFER_TAIL,OFFSET BUF1 ;
125E  C6 44 0D 00                       MOV     [SI].HANDSHAKE,0        ;handshake cntl on buf 3/4 full
                                                                        ;set up SER2 buffer:
1262  BE 0525 R                         MOV     SI,OFFSET SER2_DATA     ;point to data area
1265  C7 44 02 ---- R                   MOV     [SI].BUFFER_SEG,SEG IODATA ;
126A  C7 44 04 0572 R                   MOV     [SI].BUFFER_START,OFFSET BUF2 ;init buffer pointers
126F  C7 44 06 0592 R                   MOV     [SI].BUFFER_END,OFFSET BUF2_END;
1274  C7 44 08 0572 R                   MOV     [SI].BUFFER_HEAD,OFFSET BUF2 ;
1279  C7 44 0A 0572 R                   MOV     [SI].BUFFER_TAIL,OFFSET BUF2 ;
127E  C6 44 0D 01                       MOV     [SI].HANDSHAKE,1        ;handshake control on buffer 3/4 full
                                                                        ;set up SER3 buffer:
1282  BE 0534 R                         MOV     SI,OFFSET SER3_DATA     ;point to data area
1285  C7 44 02 ---- R                   MOV     [SI].BUFFER_SEG,SEG IODATA ;
128A  C7 44 04 0592 R                   MOV     [SI].BUFFER_START,OFFSET BUF3 ;init buffer pointers
128F  C7 44 06 05B2 R                   MOV     [SI].BUFFER_END,OFFSET BUF3_END;
1294  C7 44 08 0592 R                   MOV     [SI].BUFFER_HEAD,OFFSET BUF3 ;
1299  C7 44 0A 0592 R                   MOV     [SI].BUFFER_TAIL,OFFSET BUF3 ;
129E  C6 44 0D 01                       MOV     [SI].HANDSHAKE,1        ;handshake control on buffer 3/4 full
                                                                        ;set up SER4 buffer:
12A2  BE 0543 R                         MOV     SI,OFFSET SER4_DATA     ;point to data area
12A5  C7 44 02 ---- R                   MOV     [SI].BUFFER_SEG,SEG IODATA ;
12AA  C7 44 04 05B2 R                   MOV     [SI].BUFFER_START,OFFSET BUF4 ;init buffer pointers
12AF  C7 44 06 05D2 R                   MOV     [SI].BUFFER_END,OFFSET BUF4_END;
12B4  C7 44 08 05B2 R                   MOV     [SI].BUFFER_HEAD,OFFSET BUF4 ;
12B9  C7 44 0A 05B2 R                   MOV     [SI].BUFFER_TAIL,OFFSET BUF4 ;
12BE  C6 44 0D 01                       MOV     [SI].HANDSHAKE,1        ;handshake control on buffer 3/4 full
12C2  C3                                RET                             ;
                                BUFINIT         ENDP                    ;

                                ;*************************************************************************
                                ;* MOVEADDR                                                              *
```

```
                        ;*          Moves address CS:BX to location DS:[DI]                          *
                        ;*          Used by SCC initialization to initialize interrupt vectors       *
                        ;*                                                                           *
                        ;*Entry:    Addresses in CS:BX, DS:DI                                        *
                        ;*Exit:     none                                                             *
                        ;*Alters:   FLAGS                                                            *
                        ;*Calls:    none                                                             *
                        ;*Stack:    01H word                                                         *
                        ;*****************************************************************************
                                    ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
12C3                    MOVEADDR    PROC    NEAR                    ;moves CS:BX to DS:[DI]
12C3  89 1D                         MOV     [DI],BX                 ;move offset of routine
12C5  83 C7 02                      ADD     DI,2                    ;
12C8  8C 0D                         MOV     [DI],CS                 ;move segment
12CA  C3                            RET                             ;
                        MOVEADDR    ENDP                            ;

                                    SUBTTL  Software Interrupt Routines

                        ;**** INT 71 ***************************************************************
                        ;* SCCBIOS                                                                  *
                        ;*          This routine provides byte stream I/O to SCC communications     *
                        ;*          ports.                                                          *
                        ;*                                                                          *
                        ;* Entry:   BL = code which indicates device                                *
                        ;*               0=SER1                                                     *
                        ;*               1=SER2                                                     *
                        ;*               2=SER3                                                     *
                        ;*               3=SER4                                                     *
                        ;*          AH = Command                                                    *
                        ;*          AH = 0  Initialize the SCC communications port.  ES:DI          *
                        ;*                  points to first byte in initialization string, to       *
                        ;*                  be written to SCC command port CX contains the          *
                        ;*                  number of bytes to be written                           *
                        ;*          AH = 1  Send the character in (AL)                              *
                        ;*          AH = 2  Read a character in (AL) before returning               *
                        ;*          AH = 3  Return port status in (AX)                              *
                        ;*          AH = 4  Buffer status                                           *
                        ;*          AH = 5  Handshake line and CON reset control                    *
                        ;*                  AL = 0, hs lines DTR/RTS always enabled                 *
                        ;*                  AL = 1, DTR always enabled, RTS automatically           *
                        ;*                     disabled when input buffer is more than 3/4 full,    *
                        ;*                     reenabled when less than 3/4 full                    *
                        ;*                  AL = 2, hs lines DTR/RTS always disabled                *
                        ;*          AH = 6  Set buffer size and location.  ES:DI points to          *
                        ;*                  segment and offset of new buffer; CX contains the       *
                        ;*                  size of the new buffer in bytes                         *
                        ;*          AH = 7  Set CON reset enable                                    *
                        ;*                  AL = 1, 'ESC c' from CON causes system reset            *
                        ;*                  AL = 0, no reset from CON                               *
                        ;* Exit:    From Initialize, Return Port Status, Set Buffer, or             *
                        ;*          Handshake Line Control:                                         *
                        ;*            AH contains SCC RR1:                                          *
                        ;*                  Bit 7 = Error -- operation not completed (this bit      *
                        ;*                          is set by this routine and not by the SCC)      *
                        ;*                  Bit 6 = Framing error                                   *
                        ;*                  Bit 5 = Rcv overrun error                               *
                        ;*                  Bit 4 = Parity error                                    *
                        ;*                  Bit 3 = X (not used)                                    *
                        ;*                  Bit 2 = X (not used)                                    *
                        ;*                  Bit 1 = X (not used)                                    *
                        ;*                  Bit 0 = Shift register empty                            *
                        ;*            AL contains SCC RR0:                                          *
                        ;*                  Bit 7 = Break detect                                    *
                        ;*                  Bit 6 = X (not used)                                    *
                        ;*                  Bit 5 = Clear to send                                   *
                        ;*                  Bit 4 = X (not used)                                    *
                        ;*                  Bit 3 = Data carrier detect                             *
                        ;*                  Bit 2 = Transmit buffer empty                           *
                        ;*                  Bit 1 = Baud rate generator zero count                  *
                        ;*                  Bit 0 = Receive character available (in buffer,         *
                        ;*                          this bit is set by this routine and not by      *
                        ;*                          the SCC)                                        *
                        ;*             During initialization, data sent to WR5 is stored in RAM     *
                        ;*             data area at location WR5DATA                                *
                        ;*          From Send:                                                      *
                        ;*                                                                          *
```

```
                        ;*              Bit 7 of AH is set if routine was unable to transmit data *
                        ;*                    byte; remainder of AX is set as in port status. If *
                        ;*                    bit 7 is clear, bits 6-0 are set as in status      *
                        ;*                    request; AL is preserved.                          *
                        ;*              From Receive:                                            *
                        ;*                AL = received character.                               *
                        ;*                AH = same as in status request.  If Bit 7 is set (time *
                        ;*                     out error) AL is set as in status request.        *
                        ;*              From Buffer status:                                      *
                        ;*                AH = 0 -- no characters in buffer                      *
                        ;*                AH = 1 -- character available; AL = next character to be *
                        ;*                     read, character remains in buffer                 *
                        ;* Alters:      AX, FLAGS, as noted above                                *
                        ;* Calls:       WAIT_FOR_STATUS                                          *
                        ;* Stack:       0CH words                                                *
                        ;*****************************************************************************
12CB                                    ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
12CB                    SCCBIOS         PROC    FAR                     ;
12CB 06                                 PUSH    ES                      ;
12CC 1E                                 PUSH    DS                      ;save registers
12CD 52                                 PUSH    DX                      ;
12CE 56                                 PUSH    SI                      ;
12CF 57                                 PUSH    DI                      ;
12D0 51                                 PUSH    CX                      ;
12D1 53                                 PUSH    BX                      ;
12D2 FA                                 CLI                             ;
12D3 80 FB 03                           CMP     BL,03H                  ;disable interrupts
12D6 77 50                              JA      ERR_RETURN              ;jump if invalid device
12D8 E8 03F1 R                          CALL    DDS                     ;   (not 0 through 3)
                                                                        ;get data segment
12DB BE 0516 R                          ASSUME  DS:IODATA               ;
12DE 0A DB                              MOV     SI,OFFSET SER1_DATA     ;get offset of RAM data
12E0 74 07                              OR      BL,BL                   ;   table in SI
12E2 83 C6 0F                           JZ      SB1                     ;
12E5 FE CB                      SB2:    ADD     SI,(SIZE SERIAL_TABLE)  ;
12E7 75 F9                              DEC     BL                      ;
12E9 8B 14                              JNZ     SB2                     ;
12EB 83 FA 00                   SB1:    MOV     DX,[SI].PORT_ADDRESS    ;see if device initialized
12EE 74 38                              CMP     DX,0                    ;   (port add is not 0000H)
12F0 0A E4                              JE      ERR_RETURN              ;
12F2 75 03                              OR      AH,AH                   ;initialize if AH=0
12F4 E8 3C 90                           JNZ     TEST1                   ;
12F7 FE CC                              JMP     INITIALIZE              ;
12F9 75 03                      TEST1:  DEC     AH                      ;send char in AL if AH=1
12FB E9 1385 R                          JNZ     TEST2                   ;
12FE FE CC                              JMP     SEND                    ;
1300 75 03                      TEST2:  DEC     AH                      ;read char into AL if AH=2
1302 E9 13D5 R                          JNZ     TEST3                   ;
1305 FE CC                              JMP     READ                    ;
1307 75 03                      TEST3:  DEC     AH                      ;get port status if AH=3
1309 EB 55 90                           JNZ     TEST4                   ;
130C FE CC                              JMP     PORT_STATUS             ;
130E 75 03                      TEST4:  DEC     AH                      ;get bufr status if AH=4
1310 E9 1425 R                          JNZ     TEST5                   ;
1313 FE CC                              JMP     BUFFER_STATUS           ;
1315 75 03                      TEST5:  DEC     AH                      ;set handshake cntl if AH=5
1317 E9 143A R                          JNZ     TEST6                   ;
131A FE CC                              JMP     HS_CNTRL                ;
131C 75 03                      TEST6:  DEC     AH                      ;set new buffer if AH=6
131E E9 147F R                          JNZ     TEST7                   ;
1321 FE CC                              JMP     SET_BUFFER              ;
1323 75 03                      TEST7:  DEC     AH                      ;set CON rst cntl byte if AH=7
1325 E9 1493 R                          JNZ     ERR_RETURN              ;
1328 B4 80                              JMP     RESET_CNTRL             ;
132A 5B                         ERR_RETURN: MOV AH,080H                 ;bit 7 indicates error
132B 59                         RETURN: POP     BX                      ;restore registers
132C 5F                                 POP     CX                      ;
132D 5E                                 POP     DI                      ;
132E 5A                                 POP     SI                      ;
132F 1F                                 POP     DX                      ;
1330 07                                 POP     DS                      ;
1331 CF                                 POP     ES                      ;
                                        IRET                            ;return from SCC_IO

1332 87 F7                      INITIALIZE: XCHG SI,DI                  ;
1334 B3 00                              MOV     BL,0                    ;BL: bit 7=DF, bit 0=WF
1336 26: AC                     IN1:    LODS    ES:BYTE PTR[SI]         ;get byte
```

```
1338 80 FB 00                          CMP     BL,0            ;jump if WF/DF are clear
1338 74 0D                             JZ      IN2             ;(not data to WR1-WR15)
133D 80 E3 7F                          AND     BL,07FH         ;clear DF
1340 74 15                             JZ      IN3             ;if WF clear, send char
1342 83 0D                             MOV     BL,0            ;clear WF
1344 88 45 0E                          MOV     [DI].WR5COPY,AL ;store this WR5 data
1347 EB 0E 90                          JMP     IN3             ;
134A 3C 0F                   IN2:      CMP     AL,15           ;jump if char >15
134C 77 09                             JA      IN3             ;  ...it's a WR0 byte
134E B3 80                             MOV     BL,080H         ;it's a pointer - set DF
1350 3C 05                             CMP     AL,5            ;pointer to WR5?
1352 75 03                             JNE     IN3             ;if not send character
1354 80 CB 01                          OR      BL,001H         ;pointer to WR5 - set WF
1357                         IN3:      SCCOUT                  ;write it to command port
1357 EE                      1         OUT     DX,AL           ;SCC recovery
1358 87 D2                   1         XCHG    DX,DX           ;
135A 87 D2                   1         XCHG    DX,DX           ;
135C E2 D8                             LOOP    IN1             ;loop until all bytes sent
135E 87 FE                             XCHG    DI,SI           ;fall thru to PORT_STATUS

1360 B0 01                   PORT_STATUS: MOV   AL,1            ;point to RR1
                                       SCCOUT                  ;
1362 EE                      1         OUT     DX,AL           ;SCC recovery
1363 87 D2                   1         XCHG    DX,DX           ;
1365 87 D2                   1         XCHG    DX,DX           ;
                                       SCCIN                   ;get SCC RR1
1367 EC                      1         IN      AL,DX           ;SCC recovery
1368 87 D2                   1         XCHG    DX,DX           ;
136A 87 D2                   1         XCHG    DX,DX           ;
136C 24 7F                             AND     AL,07FH         ;clear bit 7, error bit
136E 8A E0                             MOV     AH,AL           ;move to AH
                                       SCCIN                   ;get SCC RR0 in AL
1370 EC                      1         IN      AL,DX           ;SCC recovery
1371 87 D2                   1         XCHG    DX,DX           ;
1373 87 D2                   1         XCHG    DX,DX           ;
1375 8B 7C 08                          MOV     DI,[SI].BUFFER_HEAD ;get head of buffer pointer
1378 3B 7C 0A                          CMP     DI,[SI].BUFFER_TAIL ;compare to buffer tail
137B 75 04                             JNZ     PS1             ;jump - char in buffer
137D 24 FE                             AND     AL,0FEH         ;clear char available bit
137F EB A9                             JMP     RETURN          ;
1381 0C 01                   PS1:      OR      AL,01H          ;set char available bit
1383 EB A5                             JMP     RETURN          ;

1385 50                      SEND:     PUSH    AX              ;save char to send
1386 B7 2C                             MOV     BH,02CH         ;mask - DCD, CTS, buf empty
                                       SCCIN                   ;check RR0
1388 EC                      1         IN      AL,DX           ;SCC recovery
1389 87 D2                   1         XCHG    DX,DX           ;
138B 87 D2                   1         XCHG    DX,DX           ;
138D 24 08                             AND     AL,008H         ;
138F 75 02                             JNZ     CHECK_ALL       ;if DCD is inactive:
1391 B7 04                             MOV     BH,004H         ;   check buf empty only
1393 E8 1499 R               CHECK_ALL: CALL    WAIT_FOR_STATUS ;are all true?
1396 75 26                             JNZ     NO_SEND         ;jump if not
1398 81 F2 0001                        XOR     DX,0001H        ;point to SCC data port
139C 59                                POP     CX              ;retrieve char to CL
139D 8A C1                             MOV     AL,CL           ;character in AL
                                       SCCOUT                  ;send it!
139F EE                      1         OUT     DX,AL           ;SCC recovery
13A0 87 D2                   1         XCHG    DX,DX           ;
13A2 87 D2                   1         XCHG    DX,DX           ;
13A4 81 F2 0001                        XOR     DX,0001H        ;point back to control port
13A8 B0 01                             MOV     AL,1            ;get RR1 data
                                       SCCOUT                  ;
13AA EE                      1         OUT     DX,AL           ;SCC recovery
13AB 87 D2                   1         XCHG    DX,DX           ;
13AD 87 D2                   1         XCHG    DX,DX           ;
                                       SCCIN                   ;
13AF EC                      1         IN      AL,DX           ;SCC recovery
13B0 87 D2                   1         XCHG    DX,DX           ;
13B2 87 D2                   1         XCHG    DX,DX           ;
13B4 8A E0                             MOV     AH,AL           ;RR1 status to AH
13B6 80 E4 7F                          AND     AH,07FH         ;clear bit 7, error bit
13B9 8A C1                             MOV     AL,CL           ;restore character to AL
13BB E9 132A R                         JMP     RETURN          ;
13BE 59                      NO_SEND:  POP     CX              ;adjust stack
```

```
13BF  8A E0                          MOV     AH,AL                   ;save RR0 data in AH
13C1  B0 01                          MOV     AL,1                    ;get RR1 data
                                     SCCOUT                          ;
13C3  EE                       1     OUT     DX,AL                   ;SCC recovery
13C4  87 D2                    1     XCHG    DX,DX                   ;
13C6  87 D2                    1     XCHG    DX,DX                   ;
                                     SCCIN                           ;
13C8  EC                       1     IN      AL,DX                   ;SCC recovery
13C9  87 D2                    1     XCHG    DX,DX                   ;
13CB  87 D2                    1     XCHG    DX,DX                   ;
13CD  86 C4                          XCHG    AL,AH                   ;data in correct registers
13CF  80 CC 80               TIME_OUT: OR     AH,080H                 ;indicate timeout
13D2  E9 132A R                      JMP     RETURN                  ;

13D5  8B 7C 08               READ:   MOV     DI,[SI].BUFFER_HEAD     ;get pointer
13D8  3B 7C 0A                       CMP     DI,[SI].BUFFER_TAIL     ;compare pointers
13DB  FB                             STI                             ;
13DC  74 F7                          JZ      READ                    ;wait until something is in
13DE  FA                             CLI                             ;     buffer
13DF  8E 44 02                       MOV     ES,[SI].BUFFER_SEG      ;
                                     ASSUME  ES:NOTHING              ;
13E2  26: 8A 05                      MOV     AL,ES:[DI]              ;read character from buffer
13E5  47                             INC     DI                      ;increment buf head pointer
13E6  3B 7C 06                       CMP     DI,[SI].BUFFER_END      ;at end of buffer?
13E9  75 03                          JNE     R2                      ;no, continue
13EB  8B 7C 04                       MOV     DI,[SI].BUFFER_START    ;reset buf head pointer
13EE  89 7C 08               R2:     MOV     [SI].BUFFER_HEAD,DI     ;store pointer
13F1  B4 00                          MOV     AH,00H                  ;clear error bit in AH
13F3  8A 5C 0D                       MOV     BL,[SI].HANDSHAKE       ;can handshake lines be
13F6  0A DB                          OR      BL,BL                   ;     changed?
13F8  74 28                          JZ      R3                      ;if not jump
13FA  8A 5C 0E                       MOV     BL,[SI].WR5COPY         ;check RTS handshake line
13FD  80 E3 02                       AND     BL,002H                 ;
1400  75 20                          JNZ     R3                      ;return - line enabled
1402  E8 1672 R                      CALL    BUFFER_CHECK            ;buffer > 3/4 full?
1405  75 1B                          JNZ     R3                      ;hs line should stay off
1407  8A 5C 0E                       MOV     BL,[SI].WR5COPY         ;turn on RTS handshake line
140A  80 CB 02                       OR      BL,002H                 ;
140D  88 5C 0E                       MOV     [SI].WR5COPY,BL         ;store new WR5
1410  86 F8                          XCHG    BH,AL                   ;save character in BH
1412  B0 05                          MOV     AL,5                    ;point to WR5
                                     SCCOUT                          ;output to SCC WR5
1414  EE                       1     OUT     DX,AL                   ;SCC recovery
1415  87 D2                    1     XCHG    DX,DX                   ;
1417  87 D2                    1     XCHG    DX,DX                   ;
1419  8A C3                          MOV     AL,BL                   ;
                                     SCCOUT                          ;
141B  EE                       1     OUT     DX,AL                   ;SCC recovery
141C  87 D2                    1     XCHG    DX,DX                   ;
141E  87 D2                    1     XCHG    DX,DX                   ;
1420  86 C7                          XCHG    AL,BH                   ;character back to AL
1422  E9 132A R               R3:     JMP     RETURN                  ;return

1425  32 E4                  BUFFER_STATUS: XOR    AH,AH             ;
1427  8B 7C 08                       MOV     DI,[SI].BUFFER_HEAD     ;get pointer to head of buf
142A  3B 7C 0A                       CMP     DI,[SI].BUFFER_TAIL     ;compare to buffer tail
142D  74 02                          JE      BS1                     ;
142F  FE C4                          INC     AH                      ;indicate char available
1431  8E 44 02               BS1:    MOV     ES,[SI].BUFFER_SEG      ;get character
                                     ASSUME  ES:NOTHING              ;
1434  26: 8A 05                      MOV     AL,ES:[DI]              ;
1437  E9 132A R                      JMP     RETURN                  ;

143A  3C 00                  HS_CNTRL:       CMP     AL,0            ;is AL = 0?
143C  75 0B                          JNZ     HS1                     ;no - jump
143E  88 44 0D                       MOV     [SI].HANDSHAKE,AL       ;disable auto hs control
1441  B0 82                          MOV     AL,082H                 ;add DTR/RTS active bits
1443  0A 44 0E                       OR      AL,[SI].WR5COPY         ;   to WR5COPY
1446  EB 1E 90                      JMP     HS3                     ;
1449  3C 01                  HS1:    CMP     AL,1                    ;is AL = 1?
144B  75 0B                          JNZ     HS2                     ;no - jump
144D  88 44 0D                       MOV     [SI].HANDSHAKE,AL       ;enable auto hs control
1450  B0 82                          MOV     AL,082H                 ;make sure DTR/RTS are on
1452  0A 44 0E                       OR      AL,[SI].WR5COPY         ;   (add bits to WR5COPY)
1455  EB 0F 90                      JMP     HS3                     ;
1458  3C 02                  HS2:    CMP     AL,2                    ;is AL = 2?
```

```
145A  75 20                          JNZ     HS4                     ;error: AL was not 0,1, or 2
145C  B0 00                          MOV     AL,0                    ;disable auto hs control
145E  88 44 0D                       MOV     [SI].HANDSHAKE,AL       ;
1461  B0 7D                          MOV     AL,07DH                 ;set DTR/RTS bits inactive
1463  22 44 0E                       AND     AL,[SI].WR5COPY         ;   in WR5COPY
1466  88 44 0E                HS3:   MOV     [SI].WR5COPY,AL         ;write new WR5 to WR5COPY
1469  8A E0                          MOV     AH,AL                   ;write new WR5 to SCC
146B  B0 05                          MOV     AL,5                    ;
                                     SCCOUT                          ;
146D  EE                       1     OUT     DX,AL                   ;SCC recovery
146E  87 D2                    1     XCHG    DX,DX                   ;
1470  87 D2                    1     XCHG    DX,DX                   ;
1472  8A C4                          MOV     AL,AH                   ;
                                     SCCOUT                          ;
1474  EE                       1     OUT     DX,AL                   ;SCC recovery
1475  87 D2                    1     XCHG    DX,DX                   ;
1477  87 D2                    1     XCHG    DX,DX                   ;
1479  E9 1360 R                      JMP     PORT_STATUS             ;return port status
147C  E9 1328 R                HS4:   JMP     ERR_RETURN              ;return with error bit set

147F  8C 44 02                 SET_BUFFER:   MOV     [SI].BUFFER_SEG,ES      ;set new buffer pointers
1482  89 7C 04                       MOV     [SI].BUFFER_START,DI    ;   in RAM data area
1485  89 7C 08                       MOV     [SI].BUFFER_HEAD,DI     ;   and clear buffer
1488  89 7C 0A                       MOV     [SI].BUFFER_TAIL,DI     ;
148B  03 F9                          ADD     DI,CX                   ;
148D  89 7C 06                       MOV     [SI].BUFFER_END,DI      ;
1490  E9 1360 R                      JMP     PORT_STATUS             ;return port status

1493  A2 05D8 R                RESET_CNTRL:  MOV     CON_RESET_ENABLE,AL     ;set or clear 'ESC c' reset
1496  E9 1360 R                      JMP     PORT_STATUS             ;   from console
                               SCCBIOS       ENDP                    ;
```

```
;*************************************************************************
;*                                                                       *
;* WAIT_FOR_STATUS                                                       *
;*           This subroutine looks for SCC RR0 status bits; it is called *
;*           by the SCCBIOS interrupt.                                   *
;*                                                                       *
;* Entry:    BH=bit(s) to look for, DX=SCC control port address          *
;* Exit:     AH=RR0 data, Z1=status found, Z0=timeout                    *
;* Alters:   AH, BL, CX, FLAGS                                           *
;* Calls:    None                                                        *
;* Stack:    01H word                                                    *
;*************************************************************************
                               ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
1499                           WAIT_FOR_STATUS       PROC NEAR               ;
1499  FB                               STI                             ;Enable interrupts while waiting
149A  8A 5C 0C                         MOV     BL,[SI].TIMEOUT         ;Load outer loop count
149D  2B C9                    WFS0:    SUB     CX,CX                   ;
149F                           WFS1:    SCCIN                           ;Read RR0
149F  EC                       1        IN      AL,DX                   ;SCC recovery
14A0  87 D2                    1        XCHG    DX,DX                   ;
14A2  87 D2                    1        XCHG    DX,DX                   ;
14A4  8A E0                             MOV     AH,AL                   ;Save in AH
14A6  22 C7                             AND     AL,BH                   ;Isolate bits to test
14A8  3A C7                             CMP     AL,BH                   ;Status bits the same?
14AA  74 08                             JZ      WFS_END                 ;Return with Z1 if same
14AC  E2 F1                             LOOP    WFS1                    ;Try again until CX=0...
14AE  FE CB                             DEC     BL                      ;
14B0  75 EB                             JNZ     WFS0                    ;...and again until BL=0
14B2  0A FF                             OR      BH,BH                   ;Clear Z (indicate timeout)
14B4  FA                       WFS_END: CLI                             ;Disable interrupts again
14B5  C3                                RET                             ;Return
                               WAIT_FOR_STATUS       ENDP                    ;
```

```
;*************************************************************************
;*                                                                       *
;* WRITE_TTY                                                             *
;*           This routine handles the "special" trap 29 for the console. *
;*           Outputs character in AL to console.                         *
;*                                                                       *
;* Entry:    AL = character to be sent                                   *
;* Exit:     none                                                        *
;* Alters:   none                                                        *
;* Calls:    SCC_TRAP                                                    *
;* Stack:    11H words                                                   *
;*************************************************************************
                               ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
```

```
14B6                            WRITE_TTY       PROC    FAR                     ;
14B6  1E                                        PUSH    DS                      ;
14B7  50                                        PUSH    AX                      ;
14B8  53                                        PUSH    BX                      ;
14B9  52                                        PUSH    DX
14BA  E8 03F1 R                                 CALL    DDS                     ;set DS = IODATA
                                                ASSUME  DS:IODATA               ;
14BD  8B 1E 048B R                              MOV     BX,RES_BIOS_FLAG        ;get resi bios flag
14C1  80 E3 01                                  AND     BL,01H                  ;
14C4  75 1E                                     JNZ     WRTOVIDEO               ;if EGA KBD, go write to it
14C6  8A 3E 05D7 R                              MOV     BH,SCCFLAG              ;get SCC flag
14CA  80 FF 00                                  CMP     BH,0                    ;
14CD  74 0C                                     JZ      WRTO@@CON               ;if no SCC, send char to COM1
14CF  B4 01                                     MOV     AH,SCC_SEND             ;
14D1  8B 0000                                   MOV     BX,CON_DEV_NUM          ;load SCC device number
14D4  CD 71                                     INT     SCC_TRAP                ;send char to SCC device
14D6  5A                        XITWRTTY:       POP     DX
14D7  5B                                        POP     BX                      ;
14D8  58                                        POP     AX                      ;
14D9  1F                                        POP     DS                      ;
14DA  CF                                        IRET                            ;

14DB  B4 69                     WRTO@@CON:      MOV     AH,XMIT_@@CON           ;set AH = @@CON write func
14DD  8A 0000                                   MOV     DX,CON_DEV_NUM          ;set DX = device number
14E0  CD 14                                     INT     COM1_TRAP               ;
14E2  EB F2                                     JMP     SHORT XITWRTTY          ;

14E4  8B 1E 06A6 R              WRTOVIDEO:      MOV     BX,CLS_PTR
14E8  2E: 3A 07                                 CMP     AL,CS:[BX]
14EB  75 17                                     JNE     RST_CLS_PTR
14ED  43                                        INC     BX
14EE  2E: 80 3F 00                              CMP     BYTE PTR CS:[BX],00H
14F2  75 13                                     JNE     SAVE_CLS_PTR
14F4  B4 0F                                     MOV     AH,VIDEO_STATE
14F6  CD 10                                     INT     VIDEO_INT
14F8  B4 00                                     MOV     AH,VIDEO_MODE
14FA  CD 10                                     INT     VIDEO_INT
14FC  C7 06 06A6 R 1514 R                       MOV     CLS_PTR,OFFSET CLS_STRING
1502  EB D2                                     JMP     SHORT XITWRTTY

1504  BB 1514 R                 RST_CLS_PTR:    MOV     BX,OFFSET CLS_STRING
1507  89 1E 06A6 R              SAVE_CLS_PTR:   MOV     CLS_PTR,BX
150B  B4 0E                                     MOV     AH,VIDEO_TTY            ;set AH = Video write TTY func
150D  B8 0007                                   MOV     BX,7                    ;set attributes = none
1510  CD 10                                     INT     VIDEO_INT               ;write to video card
1512  EB C2                                     JMP     SHORT XITWRTTY          ;exit


1514  1B 5B 32 4A 00            CLS_STRING      DB      ESC,'[2J',0

                                WRITE_TTY       ENDP                            ;

                                                SUBTTL  Hardware Interrupt Routines

                                ;*********************************************************************
                                ;* SER1_INT, SER2_INT, SER3_INT, SER4_INT                            *
                                ;*           These are the hardware interrupt service routines for the *
                                ;*           SCC character devices' receive character interrupts.  They *
                                ;*           set up their own stack, load a pointer to their respective *
                                ;*           RAM data areas, and then jump to a common routine (RCV_SVC) *
                                ;*           which actually services the interrupt.                  *
                                ;*                                                                   *
                                ;* Entry:    none                                                    *
                                ;* Exit:     none                                                    *
                                ;* Alters:   none                                                    *
                                ;* Calls:    RCV_SVC                                                 *
                                ;* Stack:    05H words on current stack, 08H words on INTSTACK       *
                                ;*********************************************************************
                                                ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
1519                            SER1_INT        PROC    FAR                     ;
                                                ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
1519  1E                                        PUSH    DS                      ;save registers
151A  50                                        PUSH    AX                      ;
151B  B8 ---- R                                 MOV     AX,SEG IODATA           ;point DS to RAM area
151E  8E D8                                     MOV     DS,AX                   ;
                                                ASSUME  DS:IODATA               ;
```

```
1520  8C 16 069F R                    MOV     INTR_SS,SS              ;save current stack pointer
1524  89 26 069D R                    MOV     INTR_SP,SP              ;     and segment
1528  8E D0                          MOV     SS,AX                   ;     .
152A  8D 26 0699 R                    LEA     SP,INTSTACK             ;
152E  56                              PUSH    SI                      ;
152F  BE 0516 R                       MOV     SI,OFFSET SER1_DATA     ;table address in RAM
1532  E8 15B9 R                       CALL    RCV_SVC                 ;
1535  5E                              POP     SI                      ;
1536  8E 16 069F R                    MOV     SS,INTR_SS              ;restore stack
153A  8B 26 069D R                    MOV     SP,INTR_SP              ;
153E  58                              POP     AX                      ;restore registers
153F  1F                              POP     DS                      ;
1540  CF                              IRET                            ;
                              SER1_INT        ENDP                    ;

1541                          SER2_INT        PROC    FAR             ;
                                              ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
1541  1E                                      PUSH    DS              ;save registers
1542  50                                      PUSH    AX              ;
1543  B8 ---- R                               MOV     AX,SEG IODATA   ;point DS to RAM area
1546  8E D8                                   MOV     DS,AX           ;
                                              ASSUME  DS:IODATA       ;
1548  8C 16 069F R                            MOV     INTR_SS,SS      ;save current stack pointer
154C  89 26 069D R                            MOV     INTR_SP,SP      ;     and segment
1550  8E D0                                   MOV     SS,AX           ;
1552  8D 26 0699 R                            LEA     SP,INTSTACK     ;
1556  56                                      PUSH    SI              ;
1557  BE 0525 R                               MOV     SI,OFFSET SER2_DATA     ;table address in RAM
155A  E8 15B9 R                               CALL    RCV_SVC         ;
155D  5E                                      POP     SI              ;
155E  8E 16 069F R                            MOV     SS,INTR_SS      ;restore stack
1562  8B 26 069D R                            MOV     SP,INTR_SP      ;
1566  58                                      POP     AX              ;restore registers
1567  1F                                      POP     DS              ;
1568  CF                                      IRET                    ;
                                      SER2_INT        ENDP            ;

1569                                  SER3_INT        PROC    FAR     ;
                                                      ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
1569  1E                                              PUSH    DS      ;save registers
156A  50                                              PUSH    AX      ;
156B  B8 ---- R                                       MOV     AX,SEG IODATA   ;point DS to RAM area
156E  8E D8                                           MOV     DS,AX   ;
                                                      ASSUME  DS:IODATA       ;
1570  8C 16 069F R                                    MOV     INTR_SS,SS      ;save current stack pointer
1574  89 26 069D R                                    MOV     INTR_SP,SP      ;     and segment
1578  8E D0                                           MOV     SS,AX           ;
157A  8D 26 0699 R                                    LEA     SP,INTSTACK     ;
157E  56                                              PUSH    SI              ;
157F  BE 0534 R                                       MOV     SI,OFFSET SER3_DATA     ;table address in RAM
1582  E8 15B9 R                                       CALL    RCV_SVC         ;
1585  5E                                              POP     SI              ;
1586  8E 16 069F R                                    MOV     SS,INTR_SS      ;restore stack
158A  8B 26 069D R                                    MOV     SP,INTR_SP      ;
158E  58                                              POP     AX              ;restore registers
158F  1F                                              POP     DS              ;
1590  CF                                              IRET                    ;
                                              SER3_INT        ENDP            ;

1591                                          SER4_INT        PROC    FAR     ;
                                                              ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
1591  1E                                                      PUSH    DS      ;save registers
1592  50                                                      PUSH    AX      ;
1593  B8 ---- R                                               MOV     AX,SEG IODATA   ;point DS to RAM area
1596  8E D8                                                   MOV     DS,AX   ;
                                                              ASSUME  DS:IODATA       ;
1598  8C 16 069F R                                            MOV     INTR_SS,SS      ;save current stack pointer
159C  89 26 069D R                                            MOV     INTR_SP,SP      ;     and segment
15A0  8E D0                                                   MOV     SS,AX           ;
15A2  8D 26 0699 R                                            LEA     SP,INTSTACK     ;
15A6  56                                                      PUSH    SI              ;
15A7  BE 0543 R                                               MOV     SI,OFFSET SER4_DATA     ;table address in RAM
15AA  E8 15B9 R                                               CALL    RCV_SVC         ;
15AD  5E                                                      POP     SI              ;
15AE  8E 16 069F R                                            MOV     SS,INTR_SS      ;restore stack
15B2  8B 26 069D R                                            MOV     SP,INTR_SP      ;
```

```
1586  58                            POP     AX                      ;restore registers
1587  1F                            POP     DS                      ;
1588  CF                            IRET                            ;
                      SER4_INT      ENDP                            ;

                      ;*********************************************************************
                      ;* RCV_SVC                                                           *
                      ;*          This subroutine is used to service the rcv char interrupt *
                      ;*          from the SCC device whose data area is pointed to by SI.  *
                      ;*          The incoming character is placed in the input buffer if the *
                      ;*          character was received without errors and the buffer is not *
                      ;*          full.  If the handshake line control flag is set in the data*
                      ;*          area, the RTS handshake line is turned off when the buffer *
                      ;*          becomes 3/4 full.                                        *
                      ;*                                                                   *
                      ;* Entry:   SI points to data area for this character device         *
                      ;* Exit:    none                                                     *
                      ;* Alters:  AX, FLAGS                                                *
                      ;* Calls:   BUFFER_CHECK                                             *
                      ;* Stack:   07H words                                                *
                      ;*********************************************************************
                      ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
1589                  RCV_SVC PROC    NEAR                    ;
1589  06                      PUSH    ES                      ;save registers
158A  57                      PUSH    DI                      ;
158B  52                      PUSH    DX                      ;
158C  51                      PUSH    CX                      ;
158D  53                      PUSH    BX                      ;
158E  8E 44 02                MOV     ES,[SI].BUFFER_SEG      ;
                              ASSUME  ES:NOTHING              ;
15C1  8B 14                   MOV     DX,[SI].PORT_ADDRESS    ;port address to DX
15C3  B0 01                   MOV     AL,1                    ;read RR1
                              SCCOUT                          ;
15C5  EE                1     OUT     DX,AL                   ;SCC recovery
15C6  87 D2             1     XCHG    DX,DX                   ;
15C8  87 D2             1     XCHG    DX,DX                   ;
                              SCCIN                           ;
15CA  EC                1     IN      AL,DX                   ;SCC recovery
15CB  87 D2             1     XCHG    DX,DX                   ;
15CD  87 D2             1     XCHG    DX,DX                   ;
15CF  24 70                   AND     AL,070H                 ;mask all but error bits
15D1  74 03                   JZ      RT0                     ;jump if no errors
15D3  E9 165C R               JMP     RT1                     ;jump if errors
15D6  81 F2 0001        RT0:  XOR     DX,0001H                ;point to SCC data port
                              SCCIN                           ;read character from SCC
15DA  EC                1     IN      AL,DX                   ;SCC recovery
15DB  87 D2             1     XCHG    DX,DX                   ;
15DD  87 D2             1     XCHG    DX,DX                   ;
15DF  8B 5C 0A                MOV     BX,[SI].BUFFER_TAIL     ;get end pointer to buffer
15E2  8B FB                   MOV     DI,BX                   ;save pointer in DI
15E4  3C 63                   CMP     AL,"c"                  ;CON reset test: is char "c"?
15E6  75 1F                   JNZ     NO_RESET                ;   -jump if not
15E8  4F                      DEC     DI                      ;   get last char in buffer
15E9  26: 8A 25               MOV     AH,ES:[DI]              ;
15EC  47                      INC     DI                      ;
15ED  80 FC 1B                CMP     AH,01BH                 ;   was the last buffer char ESC?
15F0  75 15                   JNZ     NO_RESET                ;   -jump if not
15F2  8A 26 05D8 R            MOV     AH,CON_RESET_ENABLE     ;   ok - reset if CON_RESET_ENABLE
15F6  0A E4                   OR      AH,AH                   ;   is not 0
15F8  74 0D                   JZ      NO_RESET                ;
15FA  50                      PUSH    AX                      ;
15FB  A1 048B R               MOV     AX,RES_BIOS_FLAG        ;
15FE  25 0001                 AND     AX,0001H                ;if EGA installed, this aint
1601  58                      POP     AX                      ;the console
1602  75 03                   JNZ     NO_RESET                ;
1604  E9 003C R               JMP     BIOS                    ;   reset STD system
1607  43              NO_RESET: INC     BX                      ;increment pointer
1608  3B 5C 06                MOV     BX,[SI].BUFFER_END      ;at end of buffer?
160B  75 03                   JNE     RT2                     ;no, continue
160D  8B 5C 04                MOV     BX,[SI].BUFFER_START    ;reset pointer
1610  3B 5C 08        RT2:    CMP     BX,[SI].BUFFER_HEAD     ;buffer full?
1613  75 06                   JNE     SCC_NOT_FULL            ;NO, continue
1615  B0 07                   MOV     AL,BELL                 ;send bell to terminal
1617  CD 29                   INT     TTY_TRAP                ;
1619  EB 06                   JMP     SHORT RT3               ;
161B  26: 88 05       SCC_NOT_FULL: MOV ES:[DI],AL              ;character to buffer
```

```
161E  89 5C 0A                  MOV     [SI].BUFFER_TAIL,BX     ;save new buffer tail
1621  81 F2 0001          RT3:  XOR     DX,0001H                ;point to SCC command port
1625  B0 38              RT4:  MOV     AL,038H                 ;reset highest interrupt under service
                                SCCOUT                        ;
1627  EE                1       OUT     DX,AL                 ;SCC recovery
1628  87 D2             1       XCHG    DX,DX                 ;
162A  87 D2             1       XCHG    DX,DX                 ;
162C  B0 38                     MOV     AL,038H               ;reset highest i.u.s. again
                                SCCOUT                        ;
162E  EE                1       OUT     DX,AL                 ;SCC recovery
162F  87 D2             1       XCHG    DX,DX                 ;
1631  87 D2             1       XCHG    DX,DX                 ;
1633  8A 64 0D                   MOV     AH,[SI].HANDSHAKE     ;check to see if handshake line
1636  0A E4                     OR      AH,AH                 ;   RTS can be changed
1638  74 1C                     JZ      RT5                   ;
                                                              ;turn off handshake line if buffer
                                                              ;   is more than 3/4 full
163A  E8 1672 R                 CALL    BUFFER_CHECK          ;test for buffer > 3/4 full
163D  74 17                     JZ      RT5                   ;
163F  8A 64 0E                  MOV     AH,[SI].WR5COPY       ;get last WR5 data
1642  80 E4 FD                  AND     AH,0FDH               ;set RTS bit low
1645  88 64 0E                  MOV     [SI].WR5COPY,AH       ;store new WR5
1648  B0 05                     MOV     AL,5                  ;point to WR5
                                SCCOUT                        ;~
164A  EE                1       OUT     DX,AL                 ;SCC recovery
164B  87 D2             1       XCHG    DX,DX                 ;
164D  87 D2             1       XCHG    DX,DX                 ;
164F  8A C4                     MOV     AL,AH                 ;
                                SCCOUT                        ;turn off RTS
1651  EE                1       OUT     DX,AL                 ;SCC recovery
1652  87 D2             1       XCHG    DX,DX                 ;
1654  87 D2             1       XCHG    DX,DX                 ;
1656  5B                RT5:  POP     BX                    ;restore reg, return
1657  59                      POP     CX                    ;
1658  5A                      POP     DX                    ;
1659  5F                      POP     DI                    ;
165A  07                      POP     ES                    ;
165B  C3                      RET                           ;

165C  81 F2 0001          RT1:  XOR     DX,0001H              ;read data byte w/ error
                                SCCIN                         ;
1660  EC                1       IN      AL,DX                 ;SCC recovery
1661  87 D2             1       XCHG    DX,DX                 ;
1663  87 D2             1       XCHG    DX,DX                 ;
1665  81 F2 0001              XOR     DX,0001H              ;point to command port
1669  B0 30                    MOV     AL,030H               ;SCC error reset
                                SCCOUT                        ;
166B  EE                1       OUT     DX,AL                 ;SCC recovery
166C  87 D2             1       XCHG    DX,DX                 ;
166E  87 D2             1       XCHG    DX,DX                 ;
1670  EB B3                     JMP     RT4                   ;reset, restore and return

                        RCV_SVC ENDP                          ;

;*****************************************************************************
;*                                                                           *
;* BUFFER_CHECK                                                              *
;*          This subroutine checks to see if the input buffer for an         *
;*          SCC character device is more than 3/4 full.                      *
;*                                                                           *
;*                                                                           *
;* Entry:   SI = pointer to SCC device data area in RAM                      *
;* Exit:    ZF = 1, buffer is less than or exactly 3/4 full                  *
;*          ZF = 0, buffer is more than 3/4 full                             *
;* Alters:  CX, BX, FLAGS                                                    *
;* Calls:   None                                                             *
;* Stack:   01H word                                                         *
;*                                                                           *
;*****************************************************************************
                        ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING

1672                    BUFFER_CHECK PROC NEAR                ;
1672  8B 4C 06                  MOV     CX,[SI].BUFFER_END    ;buffer size in bytes to CX
1675  2B 4C 04                  SUB     CX,[SI].BUFFER_START  ;
1678  8B 5C 08                  MOV     BX,[SI].BUFFER_HEAD   ;subtract tail from head
167B  2B 5C 0A                  SUB     BX,[SI].BUFFER_TAIL   ;
167E  74 10                     JZ      BC1                   ;if equal, buffer is empty
1680  73 02                     JNC     BC2                   ;go on if head > tail
1682  03 D9                     ADD     BX,CX                 ;adjust if tail > head
1684  4B                BC2:  DEC     BX                    ;get # of bytes left
```

```
1685  D1 E9                     SHR     CX,1                    ;divide buffer size by 4
1687  D1 E9                     SHR     CX,1                    ;
1689  3B D9                     CMP     BX,CX                   ;compare to # of bytes left
168B  7D 01                     JGE     BC3                     ;jump if # > 1/4 buffer
168D  C3                        RET                             ;# < 1/4 buffer left, ZF=0
168E  3B DB               BC3:  CMP     BX,BX                   ;make sure ZF=1
1690  C3                  BC1:  RET                             ;
                          BUFFER_CHECK  ENDP                    ;

                          ;*************************************************************************
                          ;* SER1_BRKINT, SER2_BRKINT, SER3_BRKINT, SER4_BRKINT                    *
                          ;*           These are the hardware break interrupt service routines for *
                          ;*           the SCC character devices. Break interrupts are generated   *
                          ;*           both at the beginning and end of a break condition.  These  *
                          ;*           routines set up their own stack, load a pointer to their    *
                          ;*           respective RAM data areas, and then jump to a common routine*
                          ;*           (BRK_SVC) which actually services the interrupt.            *
                          ;*                                                                       *
                          ;* Entry:    None                                                        *
                          ;* Exit:     None                                                        *
                          ;* Alters:   None                                                        *
                          ;* Calls:    BRK_SVC                                                     *
                          ;* Stack:    05H words on current stack, 06H words on INTSTACK           *
                          ;*************************************************************************
                                    ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
1691                      SER1_BRKINT   PROC    FAR             ;
1691  1E                              PUSH    DS                ;save registers
1692  50                              PUSH    AX                ;
1693  B8 ---- R                       MOV     AX,SEG IODATA     ;point DS to RAM area
1696  8E D8                           MOV     DS,AX             ;
                                      ASSUME  DS:IODATA         ;
1698  8C 16 069F R                    MOV     INTR_SS,SS        ;save current stack pointer and segment
169C  89 26 069D R                    MOV     INTR_SP,SP        ;
16A0  8E D0                           MOV     SS,AX             ;
16A2  8D 26 0699 R                    LEA     SP,INTSTACK       ;
16A6  56                              PUSH    SI                ;
16A7  BE 0516 R                       MOV     SI,OFFSET SER1_DATA ;table address in RAM
16AA  E8 1731 R                       CALL    BRK_SVC           ;
16AD  5E                              POP     SI                ;
16AE  8E 16 069F R                    MOV     SS,INTR_SS        ;restore stack
16B2  8B 26 069D R                    MOV     SP,INTR_SP        ;
16B6  58                              POP     AX                ;restore registers
16B7  1F                              POP     DS                ;
16B8  CF                              IRET                      ;
                          SER1_BRKINT   ENDP                    ;

                                    ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
16B9                      SER2_BRKINT   PROC    FAR             ;
16B9  1E                              PUSH    DS                ;save registers
16BA  50                              PUSH    AX                ;
16BB  B8 ---- R                       MOV     AX,SEG IODATA     ;point DS to RAM area
16BE  8E D8                           MOV     DS,AX             ;
                                      ASSUME  DS:IODATA         ;
16C0  8C 16 069F R                    MOV     INTR_SS,SS        ;save current stack pointer and segment
16C4  89 26 069D R                    MOV     INTR_SP,SP        ;
16C8  8E D0                           MOV     SS,AX             ;
16CA  8D 26 0699 R                    LEA     SP,INTSTACK       ;
16CE  56                              PUSH    SI                ;
16CF  BE 0525 R                       MOV     SI,OFFSET SER2_DATA ;table address in RAM
16D2  E8 1731 R                       CALL    BRK_SVC           ;
16D5  5E                              POP     SI                ;
16D6  8E 16 069F R                    MOV     SS,INTR_SS        ;restore stack
16DA  8B 26 069D R                    MOV     SP,INTR_SP        ;
16DE  58                              POP     AX                ;restore registers
16DF  1F                              POP     DS                ;
16E0  CF                              IRET                      ;
                          SER2_BRKINT   ENDP                    ;

                                    ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
16E1                      SER3_BRKINT   PROC    FAR             ;
16E1  1E                              PUSH    DS                ;save registers
16E2  50                              PUSH    AX                ;
16E3  B8 ---- R                       MOV     AX,SEG IODATA     ;point DS to RAM area
16E6  8E D8                           MOV     DS,AX             ;
                                      ASSUME  DS:IODATA         ;
16E8  8C 16 069F R                    MOV     INTR_SS,SS        ;save current stack pointer and segment
```

```
16EC  89 26 069D R                MOV     INTR_SP,SP              ;
16F0  8E D0                       MOV     SS,AX                   ;
16F2  8D 26 0699 R                LEA     SP,INTSTACK             ;
16F6  56                          PUSH    SI                      ;
16F7  BE 0534 R                   MOV     SI,OFFSET SER3_DATA     ;table address in RAM
16FA  E8 1731 R                   CALL    BRK_SVC                 ;
16FD  5E                          POP     SI                      ;
16FE  8E 16 069F R                MOV     SS,INTR_SS              ;restore stack
1702  8B 26 069D R                MOV     SP,INTR_SP              ;
1706  58                          POP     AX                      ;restore registers
1707  1F                          POP     DS                      ;
1708  CF                          IRET                            ;
                          SER3_BRKINT     ENDP                    ;

                                  ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
1709                      SER4_BRKINT     PROC    FAR             ;
1709  1E                          PUSH    DS                      ;save registers
170A  50                          PUSH    AX                      ;
170B  B8 ---- R                   MOV     AX,SEG IODATA           ;point DS to RAM area
170E  8E D8                       MOV     DS,AX                   ;
                                  ASSUME  DS:IODATA               ;
1710  8C 16 069F R                MOV     INTR_SS,SS              ;save current stack pointer and segment
1714  89 26 069D R                MOV     INTR_SP,SP              ;
1718  8E D0                       MOV     SS,AX                   ;
171A  8D 26 0699 R                LEA     SP,INTSTACK             ;
171E  56                          PUSH    SI                      ;
171F  BE 0543 R                   MOV     SI,OFFSET SER4_DATA     ;table address in RAM
1722  E8 1731 R                   CALL    BRK_SVC                 ;
1725  5E                          POP     SI                      ;
1726  8E 16 069F R                MOV     SS,INTR_SS              ;restore stack
172A  8B 26 069D R                MOV     SP,INTR_SP              ;
172E  58                          POP     AX                      ;restore registers
172F  1F                          POP     DS                      ;
1730  CF                          IRET                            ;
                          SER4_BRKINT     ENDP                    ;

                          ;*****************************************************************
                          ;* BRK_SVC                                                       *
                          ;*       This subroutine is used to service the break interrupt from *
                          ;*       the SCC device whose data area is pointed to by SI.  At the *
                          ;*       beginning of a break, the appropriate buffer is cleared, a  *
                          ;*       ^C character is inserted at the beginning of the buffer, and*
                          ;*       the rcv interrupt is disabled.  At the end of a break, the  *
                          ;*       null character is removed from the SCC read register and the*
                          ;*       rcv interrupt is reenabled.                                 *
                          ;*                                                                   *
                          ;* Entry:  SI points to data area in IODATA for this device          *
                          ;* Exit:   none                                                      *
                          ;* Alters: AX, FLAGS                                                 *
                          ;* Calls:  SCC_FLUSH                                                 *
                          ;* Stack:  05H words                                                 *
                          ;*****************************************************************
1731                      BRK_SVC         PROC    NEAR            ;
                                  ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
1731  52                          PUSH    DX                      ;save registers
1732  57                          PUSH    DI                      ;
1733  06                          PUSH    ES                      ;
1734  8B 14                       MOV     DX,[SI].PORT_ADDRESS    ;get port address
                                  SCCIN                           ;read RR0
1736  EC                    1     IN      AL,DX                   ;SCC recovery
1737  87 D2                 1     XCHG    DX,DX                   ;
1739  87 D2                 1     XCHG    DX,DX                   ;
1738  24 80                       AND     AL,80H                  ;
1730  74 27                       JZ      BSV1                    ;jump if break is not active
173F  8E 44 02                    MOV     ES,[SI].BUFFER_SEG      ;buffer segment to ES
                                  ASSUME  ES:NOTHING              ;
1742  8B 7C 04                    MOV     DI,[SI].BUFFER_START    ;clear buffer
1745  89 7C 08                    MOV     [SI].BUFFER_HEAD,DI     ;
1748  26: C6 05 03                MOV     BYTE PTR ES:[DI],ETX    ;move ^C to buffer
174C  47                          INC     DI                      ;
174D  89 7C 0A                    MOV     [SI].BUFFER_TAIL,DI     ;
1750  B0 01                       MOV     AL,1                    ;disable rcv char interrupt
                                  SCCOUT                          ;
1752  EE                    1     OUT     DX,AL                   ;SCC recovery
1753  87 D2                 1     XCHG    DX,DX                   ;
1755  87 D2                 1     XCHG    DX,DX                   ;
```

```
1757  B0 01                             MOV     AL,01H                  ;
                                        SCCOUT                          ;
1759  EE                        1       OUT     DX,AL                   ;SCC recovery
175A  87 D2                     1       XCHG    DX,DX                   ;
175C  87 D2                     1       XCHG    DX,DX                   ;
175E  E8 1790 R                         CALL    SCC_FLUSH               ;discard any chars in SCC buf
1761  CD 1B                             INT     1BH                     ;go to user break routine
1763  EB 12 90                          JMP     BSV2                    ;reset interrupts and return
1766  E8 1790 R                 BSV1:   CALL    SCC_FLUSH               ;discard null character left from break
1769  B0 01                             MOV     AL,1                    ;enable rcv char interrupt
                                        SCCOUT                          ;
176B  EE                        1       OUT     DX,AL                   ;SCC recovery
176C  87 D2                     1       XCHG    DX,DX                   ;
176E  87 D2                     1       XCHG    DX,DX                   ;
1770  B0 11                             MOV     AL,011H                 ;
                                        SCCOUT                          ;
1772  EE                        1       OUT     DX,AL                   ;SCC recovery
1773  87 D2                     1       XCHG    DX,DX                   ;
1775  87 D2                     1       XCHG    DX,DX                   ;
1777  B0 10                     BSV2:   MOV     AL,010H                 ;reset ext/stat interrupts
                                        SCCOUT                          ;
1779  EE                        1       OUT     DX,AL                   ;SCC recovery
177A  87 D2                     1       XCHG    DX,DX                   ;
177C  87 D2                     1       XCHG    DX,DX                   ;
177E  B0 38                             MOV     AL,038H                 ;reset highest interrupt under service
                                        SCCOUT                          ;
1780  EE                        1       OUT     DX,AL                   ;SCC recovery
1781  87 D2                     1       XCHG    DX,DX                   ;
1783  87 D2                     1       XCHG    DX,DX                   ;
1785  B0 38                             MOV     AL,038H                 ;reset highest i.u.s. again
                                        SCCOUT                          ;
1787  EE                        1       OUT     DX,AL                   ;SCC recovery
1788  87 D2                     1       XCHG    DX,DX                   ;
178A  87 D2                     1       XCHG    DX,DX                   ;
178C  07                                POP     ES                      ;
178D  5F                                POP     DI                      ;restore registers
178E  5A                                POP     DX                      ;
178F  C3                                RET                             ;return
                                BRK_SVC ENDP                            ;

                                ;*************************************************************************
                                ;* SCC_FLUSH                                                             *
                                ;*              Flushes characters from an SCC's internal input buffer and  *
                                ;*              discards them.  Does not flush characters received after the*
                                ;*              null character left over from a break.                   *
                                ;*                                                                       *
                                ;* Entry:       DX=SCC control port address                              *
                                ;* Exit:        none                                                     *
                                ;* Alters:      AX, FLAGS                                                *
                                ;* Calls:       none                                                     *
                                ;* Stack:       01H word                                                 *
                                ;*************************************************************************
1790                            SCC_FLUSH       PROC    NEAR            ;
1790                            SFL2:           SCCIN                   ;character available?
1790  EC                        1               IN      AL,DX           ;SCC recovery
1791  87 D2                     1               XCHG    DX,DX           ;
1793  87 D2                     1               XCHG    DX,DX           ;
1795  24 01                                     AND     AL,01H          ;
1797  74 11                                     JZ      SFL1            ;no - return
1799  81 F2 0001                                XOR     DX,0001H        ;
                                                SCCIN                   ;read character
179D  EC                        1               IN      AL,DX           ;SCC recovery
179E  87 D2                     1               XCHG    DX,DX           ;
17A0  87 D2                     1               XCHG    DX,DX           ;
17A2  81 F2 0001                                XOR     DX,0001H        ;
17A6  3C 00                                     CMP     AL,00           ;return if null char left by
17A8  75 E6                                     JNZ     SFL2            ;   break
17AA  C3                        SFL1:           RET                     ;
                                SCC_FLUSH       ENDP                    ;

                                ;*************************************************************************
                                ;* PARITY_ERROR                                                          *
                                ;*              Interrupt routine called if NMI (parity error) occurs.   *
                                ;*              This routine simply prints a message to the console then *
                                ;*              returns to allow the operator to decide what to do.      *
                                ;*                                                                       *
```

```
                        ;* Entry:     none                                                *
                        ;* Exit:      none                                                *
                        ;* Alters:    none                                                *
                        ;* Calls:     INT SCC_TRAP                                        *
                        ;* Stack:     12H words                                           *
                        ;*****************************************************************
                                    ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
17AB                    PARITY_ERROR PROC   FAR                 ;NMI points here for parity errors
17AB  56                            PUSH    SI                  ;
17AC  53                            PUSH    BX                  ;
17AD  50                            PUSH    AX                  ;
17AE  8D 36 18BD R                  LEA     SI,PARITYMSG        ;
17B2  2E: AC            PE0:        LODS    BYTE PTR CS:[SI]    ;get the character
17B4  3C 24                         CMP     AL,'$'              ;check for end of string
17B6  74 09                         JE      PE_RET              ;
17B8  B4 01                         MOV     AH,SCC_SEND         ;
17BA  BB 0000                       MOV     BX,CON_DEV_NUM      ;load SCC device number
17BD  CD 71                         INT     SCC_TRAP            ;
17BF  EB F1                         JMP     PE0                 ;
17C1  58                PE_RET:     POP     AX                  ;
17C2  5B                            POP     BX                  ;
17C3  5E                            POP     SI                  ;
17C4  CF                IRET_INST:  IRET                        ;
                        PARITY_ERROR ENDP                       ;

                        ;*****************************************************************
                        ;* INTR_ERR                                                      *
                        ;*          All unused interrupts point to here so an error message can *
                        ;*          be displayed if an uninitialized interrupt is called.*
                        ;*                                                               *
                        ;* Entry:     none                                               *
                        ;* Exit:      none                                               *
                        ;* Alters:    none                                               *
                        ;* Calls:     none                                               *
                        ;* Stack:     19H words                                          *
                        ;*****************************************************************
                                    ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
17C5                    INTR_ERR    PROC    FAR                 ;unused interrupts point here
17C5  55                            PUSH    BP                  ;
17C6  8B EC                         MOV     BP,SP               ;
17C8  1E                            PUSH    DS                  ;
17C9  56                            PUSH    SI                  ;
17CA  50                            PUSH    AX                  ;
17CB  C5 76 02                      LDS     SI,[BP+2]           ;get address of interrupted program
17CE  8B 44 FE                      MOV     AX,DS:[SI-2]        ;get the possible INT instruction
17D1  86 E0                         XCHG    AH,AL               ;first opcode in AH, second in AL
17D3  80 FC CD                      CMP     AH,0CDH             ;check for INT xx instruction
17D6  74 06                         JE      INTR_SOFT           ;
17D8  3C CC                         CMP     AL,0CCH             ;check for INT 03 instruction
17DA  75 16                         JNE     INTR_HARD           ;
17DC  B0 03                         MOV     AL,3                ;
17DE  8D 36 18D0 R      INTR_SOFT:  LEA     SI,S_INTR_MSG       ;
17E2  E8 0487 R                     CALL    MESSAGE             ;
17E5  E8 0498 R                     CALL    HEX_OUT_BYTE        ;
17E8  8D 36 18F6 R                  LEA     SI,S_INTR_MSG2      ;
17EC  E8 0487 R                     CALL    MESSAGE             ;
17EF  EB 08 90                      JMP     INTR_ERR_RET        ;
17F2  8D 36 190B R      INTR_HARD:  LEA     SI,H_INTR_MSG       ;
17F6  E8 0487 R                     CALL    MESSAGE             ;
17F9  58                INTR_ERR_RET: POP   AX                  ;
17FA  5E                            POP     SI                  ;
17FB  1F                            POP     DS                  ;
17FC  5D                            POP     BP                  ;
17FD  CF                            IRET                        ;
                        INTR_ERR    ENDP                        ;

                                    SUBTTL  Data area
                        ;*****************************************************************
                        ;*                                                               *
                        ;* Messages, text, and data area                                 *
                        ;*                                                               *
                        ;*****************************************************************
17FE  49 4F 20 20 20 20 20  IOSYSDIR    DB      'IO      SYS'       ;file name
      20 53 59 53
1809  4D 53 44 4F 53 20 20  MSDOSDIR    DB      'MSDOS   SYS'       ;file name
      20 53 59 53
1814  50 2D 4C 20 44 49 53  MEMDISKID   DB      'P-L DISK'         ;id bytes at beginning of memory disk
      4B
```

```
= 0008                          IDSIZE      EQU   8                       ;number of id bytes to check
181C  0D 0A 07 4E 6F 20 6D      DISKERROR   DB    CR,LF,BELL,'No memory disks found.'
      65 6D 6F 72 79 20 64
      69 73 6B 73 20 66 6F
      75 6E 64 2E
1835  0D 0A 49 6E 73 74 61                  DB    CR,LF,'Install memory disk and reset the system.',CR,LF,'$'
      6C 6C 20 6D 65 6D 6F
      72 79 20 64 69 73 6B
      20 61 6E 64 20 72 65
      73 65 74 20 74 68 65
      20 73 79 73 74 65 6D
      2E 0D 0A 24
1863  0D 0A 07 49 4F 2E 53      SYSERROR    DB    CR,LF,BELL,'IO.SYS or MSDOS.SYS not found on drive A.'
      59 53 20 6F 72 20 4D
      53 44 4F 53 2E 53 59
      53 20 6E 6F 74 20 66
      6F 75 6E 64 20 6F 6E
      20 64 72 69 76 65 20
      41 2E
188F  0D 0A 49 6E 73 74 61                  DB    CR,LF,'Install system disk and reset the system.',CR,LF,'$'
      6C 6C 20 73 79 73 74
      65 6D 20 64 69 73 6B
      20 61 6E 64 20 72 65
      73 65 74 20 74 68 65
      20 73 79 73 74 65 6D
      2E 0D 0A 24
188D  0D 0A 07 50 61 72 69      PARITYMSG   DB    CR,LF,BELL,'Parity Error.'
      74 79 20 45 72 72 6F
      72 2E
18CD  0D 0A 24                  CRLF        DB    CR,LF,'$'
18D0  0D 0A 07 57 41 52 4E      S_INTR_MSG  DB    CR,LF,BELL,'WARNING:  Uninitialized interrupt $'
      49 4E 47 3A 20 20 55
      6E 69 6E 69 74 69 61
      6C 69 7A 65 64 20 69
      6E 74 65 72 72 75 70
      74 20 24
18F6  48 20 68 61 73 20 62      S_INTR_MSG2 DB    'H has been called.',CR,LF,'$'
      65 65 6E 20 63 61 6C
      6C 65 64 2E 0D 0A 24
1908  0D 0A 07 57 41 52 4E      H_INTR_MSG  DB    CR,LF,BELL,'WARNING:  An uninitialized hardware interrupt'
      49 4E 47 3A 20 20 41
      6E 20 75 6E 69 6E 69
      74 69 61 6C 69 7A 65
      64 20 68 61 72 64 77
      61 72 65 20 69 6E 74
      65 72 72 75 70 74
1938  20 68 61 73 20 62 65                  DB    ' has been called.',CR,LF,'$'
      65 6E 20 63 61 6C 6C
      65 64 2E 0D 0A 24
194F  0D 0A 53 54 44 2D 44      BIOSMSG     DB    CR,LF,'STD-DOS version 2.00'
      4F 53 20 76 65 72 73
      69 6F 6E 20 32 2E 30
      30
1965  0D 0A 28 43 29 43 6F                  DB    CR,LF,'(C)Copyright 1985, 1986, 1987 Pro-Log Corp.',CR,LF,'$'
      70 79 72 69 67 68 74
      20 31 39 38 35 2C 20
      31 39 38 36 2C 20 31
      39 38 37 20 50 72 6F
      2D 4C 6F 67 20 43 6F
      72 70 2E 0D 0A 24
1995  0D 0A 53 79 73 74 65      MEMTESTMESG DB    CR,LF,'System Memory Test in Progress...',CR,LF,'$'
      6D 20 4D 65 6D 6F 72
      79 20 54 65 73 74 20
      69 6E 20 50 72 6F 67
      72 65 73 73 2E 2E 2E
      0D 0A 24
19BB  20 4B 62 79 74 65 73      TESTEDMSG   DB    ' Kbytes of System RAM $'
      20 6F 66 20 53 79 73
      74 65 6D 20 52 41 4D
      20 24
19D2  68 61 76 65 20 62 65      ISTESTEDMESG DB   'have been tested',CR,LF,BELL,'$'
      65 6E 20 74 65 73 74
      65 64 0D 0A 07 24
19E6  0D 44 65 6C 61 79 69      DELAYMSG    DB    CR,'Delaying for Hard Disk Startup$'
      6E 67 20 66 6F 72 20
      48 61 72 64 20 44 69
```

```
                73 6B 20 53 74 61 72
                74 75 70 24
1A06  0D 20 20 20 20 20 20 20   CLEARMSG      DB      CR,'                    $'
      20 20 20 20 20 20 20 20
      20 20 20 20 20 20 20 20
      20 20 20 20 20 20 20 20
      20 20 20 24


                ;------********** CONSOLE SCC TABLE (SER1) 19.2K BAUD **********
1A26  001C      SER1TABLE       DW      28              ;word = number of bytes in table
1A28  04 44                     DB       4,044H         ;WR4, x16 clk, no parity
1A2A  03 C0                     DB       3,0C0H         ;WR3, rcv 8 bits, rcv disabled
1A2C  05 60                     DB       5,060H         ;WR5, xmt 8 bits, xmt disabled
1A2E  09 01                     DB       9,001H         ;WR9, vect includes status, status lo
1A30  0A 00                     DB      10,000H         ;WR10, misc control
1A32  0B 56                     DB      11,056H         ;WR11, rcv clk=br gen, xmt clk=br gen
1A34  0C 06                     DB      12,006H         ;WR12, low br time constant, 19.2K baud
1A36  0D 00                     DB      13,000H         ;WR13, hi br time constant, 19.2K baud
1A38  0E 02                     DB      14,002H         ;WR14, baud rate gen=pclk input
1A3A  0F 80                     DB      15,080H         ;WR15, enable break interrupt
1A3C  03 C1                     DB       3,0C1H         ;WR3, enable rcv
1A3E  05 EA                     DB       5,0EAH         ;WR5, set DTR and RTS, enable xmt
1A40  0E 03                     DB      14,003H         ;WR14, enable br gen
1A42  01 11                     DB       1,011H         ;WR1, enable rcv and brk interrupt
                                                        ;WR9 written by RE_init


                ;------********** PRINTER SCC TABLE (SER2) 2400 BAUD **********
1A44  001C      SER2TABLE       DW      28              ;word = number of bytes in table
1A46  04 44                     DB       4,044H         ;WR4, x16 clk, no parity
1A48  03 C0                     DB       3,0C0H         ;WR3, rcv 8 bits, rcv disabled
1A4A  05 60                     DB       5,060H         ;WR5, xmt 8 bits, xmt disabled
1A4C  09 01                     DB       9,001H         ;WR9, vect includes status, status lo
1A4E  0A 00                     DB      10,000H         ;WR10, misc control
1A50  0B 56                     DB      11,056H         ;WR11, rcv clk=br gen, xmt clk=br gen
1A52  0C 40                     DB      12,040H         ;WR12, low br time constant, 2400 baud
1A54  0D 00                     DB      13,000H         ;WR13, hi br time constant, 2400 baud
1A56  0E 02                     DB      14,002H         ;WR14, baud rate gen=pclk input
1A58  0F 80                     DB      15,080H         ;WR15, enable break interrupt
1A5A  03 C1                     DB       3,0C1H         ;WR3, enable rcv
1A5C  05 EA                     DB       5,0EAH         ;WR5, set DTR and RTS, enable xmt
1A5E  0E 03                     DB      14,003H         ;WR14, enable br gen
1A60  01 11                     DB       1,011H         ;WR1, enable rcv and brk interrupt
                                                        ;WR9 written by RE_init


                ;------********** SER3 & SER4 SCC TABLE 9600 BAUD **********
1A62            SER3TABLE       LABEL   WORD
1A62  001C      SER4TABLE       DW      28              ;word = number of bytes in table
1A64  04 44                     DB       4,044H         ;WR4, x16 clk, no parity
1A66  03 C0                     DB       3,0C0H         ;WR3, rcv 8 bits, rcv disabled
1A68  05 60                     DB       5,060H         ;WR5, xmt 8 bits, xmt disabled
1A6A  09 01                     DB       9,001H         ;WR9, vect includes status, status lo
1A6C  0A 00                     DB      10,000H         ;WR10, misc control
1A6E  0B 56                     DB      11,056H         ;WR11, rcv clk=br gen, xmt clk=br gen
1A70  0C 0E                     DB      12,00EH         ;WR12, low br time constant, 9600 baud
1A72  0D 00                     DB      13,000H         ;WR13, hi br time constant, 9600 baud
1A74  0E 02                     DB      14,002H         ;WR14, baud rate gen=pclk input
1A76  0F 80                     DB      15,080H         ;WR15, enable break interrupt
1A78  03 C1                     DB       3,0C1H         ;WR3, enable rcv
1A7A  05 EA                     DB       5,0EAH         ;WR5, set DTR and RTS, enable xmt
1A7C  0E 03                     DB      14,003H         ;WR14, enable br gen
1A7E  01 11                     DB       1,011H         ;WR1, enable rcv and brk interrupt
                                                        ;WR9 written by RE_init


                ;******** INT 5 ***************************************************************
                ; This routine is invoked by a INT 5 command and will print current            *
                ; Video screen if EGA installed. This happens with interupts enabled and       *
                ; if another INT 5 is issued while still printing, it will be ignored.        *
                ;*********************************************************************************
                        ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING
1A80            PRINT_SCREEN    PROC    FAR             ;
1A80  FB                        STI                     ;turn interrupts on
1A81  1E                        PUSH    DS              ;save registers
1A82  52                        PUSH    DX              ;
1A83  51                        PUSH    CX              ;
1A84  53                        PUSH    BX              ;
1A85  50                        PUSH    AX              ;
```

```
1A86  E8 03F1 R                 CALL    DDS                     ;set DS = IODATA segment
                                ASSUME  DS:IODATA               ;
1A89  A1 048B R                 MOV     AX,RES_BIOS_FLAG        ;
1A8C  25 0001                   AND     AX,1                    ;
1A8F  74 43                     JZ      EXIT_PRTSCR             ;exit if EGA/KBD not installed
1A91  80 3E 0500 R 00           CMP     STATUS_BYTE,0           ;
1A96  75 3C                     JNE     EXIT_PRTSCR             ;exit if already busy
1A98  C6 06 0500 R 01           MOV     STATUS_BYTE,1           ;
1A9D  B4 03                     MOV     AH,03H                  ;get current curs position
1A9F  CD 10                     INT     10H                     ;save curs position on stack
1AA1  52                        PUSH    DX                      ;
1AA2  B4 0F                     MOV     AH,0FH                  ;
1AA4  CD 10                     INT     10H                     ;AH=char/row;BH=active page
1AA6  8A CC                     MOV     CL,AH                   ;set CL = char count
1AA8  B5 19                     MOV     CH,25                   ;set CH = 25 rows
1AAA  BA 0000                   MOV     DX,0                    ;init curs pos = 0000
1AAD  E8 1B00 R                 CALL    PRNT_CRLF               ;send CR & LF to printer
1AB0  75 28                     JNZ     PRNT_ERROR              ;exit if error
1AB2  E8 1AE1 R         PRN_NXT_CHR:    CALL    PRINT_CHAR      ;send video char @DX to printer
1AB5  75 23                     JNZ     PRNT_ERROR              ;exit if error
1AB7  FE C2                     INC     DL                      ;incr column
1AB9  3A D1                     CMP     DL,CL                   ;
1ABB  75 F5                     JNE     PRN_NXT_CHR             ;if not max chars, continue
1ABD  B2 00                     MOV     DL,0                    ;reset cur col to 0
1ABF  FE C6                     INC     DH                      ;incr cur row
1AC1  E8 1B00 R                 CALL    PRNT_CRLF               ;send CR & LF to printer
1AC4  75 14                     JNZ     PRNT_ERROR              ;exit if error
1AC6  3A F5                     CMP     DH,CH                   ;
1AC8  75 E8                     JNE     PRN_NXT_CHR             ;if not max rows, continue
1ACA  C6 06 0500 R 00           MOV     STATUS_BYTE,0           ;reset status byte
1ACF  5A                XIT_PRTSCR:     POP     DX              ;
1AD0  B4 02                     MOV     AH,02H                  ;restore original curs position
1AD2  CD 10                     INT     10H                     ;
1AD4  58                EXIT_PRTSCR:    POP     AX              ;restore registers & exit
1AD5  5B                        POP     BX                      ;
1AD6  59                        POP     CX                      ;
1AD7  5A                        POP     DX                      ;
1AD8  1F                        POP     DS                      ;
1AD9  CF                        IRET                            ;

1ADA  C6 06 0500 R FF   PRNT_ERROR:     MOV     STATUS_BYTE,0FFH        ;set status byte = error
1ADF  EB EE                     JMP     SHORT XIT_PRTSCR        ;

                        PRINT_SCREEN    ENDP                    ;

                        ; Get video char @ position DX and print it
                        ; Return ZR if ok
                                ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
1AE1                    PRINT_CHAR      PROC    NEAR            ;
1AE1  B4 02                     MOV     AH,02H                  ;
1AE3  CD 10                     INT     10H                     ;set video position @DX
1AE5  B4 08                     MOV     AH,08H                  ;
1AE7  CD 10                     INT     10H                     ;get video char
1AE9  3C 00                     CMP     AL,0                    ;
1AEB  75 02                     JNE     PRNT_CHAR               ;if ascii, continue
1AED  B0 20                     MOV     AL,20H                  ;set char = space
1AEF  E8 1AF3 R         PRNT_CHAR:      CALL    PRINT_AL        ;send AL to printer
1AF2  C3                        RET                             ;
                        PRINT_CHAR      ENDP                    ;

                        ; Send char in AL to printer
                        ; Return ZR if ok
                                ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
1AF3                    PRINT_AL        PROC    NEAR            ;
1AF3  52                        PUSH    DX                      ;
1AF4  BA 0000                   MOV     DX,0                    ;send AL to printer
1AF7  B4 00                     MOV     AH,0                    ;
1AF9  CD 17                     INT     17H                     ;
1AFB  F6 C4 25                  TEST    AH,25H                  ;test for printer error
1AFE  5A                        POP     DX                      ;
1AFF  C3                        RET                             ;
                        PRINT_AL        ENDP                    ;

                        ; Send a CR and LF to printer
                        ; Return ZR if no error
                                ASSUME  CS:CODE,DS:IODATA,ES:NOTHING
```

```
1B00                            PRNT_CRLF   PROC    NEAR                    ;
1B00  B0 0D                                 MOV     AL,CR                   ;
1B02  E8 1AF3 R                             CALL    PRINT_AL                ;send CR to printer
1B05  75 05                                 JNZ     XIT_PRN_CRLF            ;exit if error
1B07  B0 0A                                 MOV     AL,LF                   ;
1B09  E8 1AF3 R                             CALL    PRINT_AL                ;send LF to printer
1B0C  C3                        XIT_PRN_CRLF: RET                           ;
                                PRNT_CRLF   ENDP                            ;

                                ;*************************************************************************
                                ;*
                                ;*  These video parameters will make us more compatible
                                ;*
                                ;*************************************************************************

70A4                                        ORG     070A4H
70A4                            VIDEO_PARAMS LABEL  BYTE
70A4  38 28 2D 0A 1F 06 19                  DB      38H,28H,2DH,0AH,1FH,06H,19H ;set up for 40 x 25
70AB  1C 02 07 06 07                        DB      1CH,02H,07H,06H,07H         ;
70B0  00 00 00 00                           DB      00H,00H,00H,00H             ;

70B4  71 50 5A 0A 1F 06 19                  DB      71H,50H,5AH,0AH,1FH,06H,19H ;set up for 80 x 25
70BB  1C 02 07 06 07                        DB      1CH,02H,07H,06H,07H         ;
70C0  00 00 00 00                           DB      00H,00H,00H,00H             ;

70C4  38 28 2D 0A 7F 06 64                  DB      38H,28H,2DH,0AH,7FH,06H,64H ;set up for graphics
70CB  70 02 01 06 07                        DB      70H,02H,01H,06H,07H         ;
70D0  00 00 00 00                           DB      00H,00H,00H,00H             ;

70D4  61 50 52 0F 19 06 19                  DB      61H,50H,52H,0FH,19H,06H,19H ;set up for 80 x 25 B & W
70DB  1C 02 0D 0B 0C                        DB      1CH,02H,0DH,0BH,0CH         ;
70E0  00 00 00 00                           DB      00H,00H,00H,00H             ;

70E4  0800                                  DW      2048                        ;regen length: 40 x 25
70E6  1000                                  DW      4096                        ;             : 80 x 25
70E8  4000                                  DW      16384                       ;             : graphics
70EA  4000                                  DW      16384                       ;             : graphics

70EC  28 28 50 50 28 28 50                  DB      40,40,80,80,40,40,80,80     ;columns
      50

70F4  2C 28 2D 29 2A 2E 1E                  DB      2CH,28H,2DH,29H,2AH,2EH,1EH,29H  ;mode sets table
      29

                                ;*************************************************************************
                                ;*
                                ;*  This graphics char gen table will make us more compatible
                                ;*
                                ;*************************************************************************

7A6E                                        ORG     07A6EH
7A6E                            CHAR_GEN_TBL LABEL  BYTE
7A6E  00 00 00 00 00 00 00 00               DB      00H,00H,00H,00H,00H,00H,00H,00
      00
7A76  7E 81 A5 81 BD 99 81                  DB      07EH,081H,0A5H,081H,0BDH,099H,081H,07Eh
      7E
7A7E  7E FF DB FF C3 E7 FF                  DB      07EH,0FFH,0DBH,0FFH,0C3H,0E7H,0FFH,07EH ;
      7E
7A86  6C FE FE FE 7C 38 10                  DB      06CH,0FEH,0FEH,0FEH,07CH,038H,010H,000H ;
      00
7A8E  10 38 7C FE 7C 38 10                  DB      010H,038H,07CH,0FEH,07CH,038H,010H,000H
      00
7A96  38 7C 38 FE FE 7C 38                  DB      038H,07CH,038H,0FEH,0FEH,07CH,038H,07CH
      7C
7A9E  10 10 38 7C FE 7C 38                  DB      010H,010H,038H,07CH,0FEH,07CH,038H,07CH
      7C
7AA6  00 00 18 3C 3C 18 00                  DB      000H,000H,018H,03CH,03CH,018H,000H,000H
      00
7AAE  FF FF E7 C3 C3 E7 FF                  DB      0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH
      FF
7AB6  00 3C 66 42 42 66 3C                  DB      000H,03CH,066H,042H,042H,066H,03CH,000H
      00
7ABE  FF C3 99 BD BD 99 C3                  DB      0FFH,0C3H,099H,0BDH,0BDH,099H,0C3H,0FFH
      FF
7AC6  0F 07 0F 7D CC CC CC                  DB      00FH,007H,00FH,07DH,0CCH,0CCH,0CCH,078H
      78
7ACE  3C 66 66 66 3C 18 7E                  DB      03CH,066H,066H,066H,03CH,018H,07EH,018H
      18
```

```
7AD6  3F 33 3F 30 30 70 F0      DB      03FH,033H,03FH,030H,030H,070H,0F0H,0E0H
      E0
7ADE  7F 63 7F 63 63 67 E6      DB      07FH,063H,07FH,063H,063H,067H,0E6H,0C0H
      C0
7AE6  99 5A 3C E7 E7 3C 5A      DB      099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H
      99

7AEE  80 E0 F8 FE F8 E0 80      DB      080H,0E0H,0F8H,0FEH,0F8H,0E0H,080H,000H
      00
7AF6  02 0E 3E FE 3E 0E 02      DB      002H,00EH,03EH,0FEH,03EH,00EH,002H,000H
      00
7AFE  18 3C 7E 18 18 7E 3C      DB      018H,03CH,07EH,018H,018H,07EH,03CH,018H
      18
7B06  66 66 66 66 66 00 66      DB      066H,066H,066H,066H,066H,000H,066H,000H
      00
7B0E  7F DB DB 7B 1B 1B 1B      DB      07FH,0DBH,0DBH,07BH,01BH,01BH,01BH,000H
      00
7B16  3E 63 38 6C 6C 38 CC      DB      03EH,063H,038H,06CH,06CH,038H,0CCH,078H
      78
7B1E  00 00 00 00 7E 7E 7E      DB      000H,000H,000H,000H,07EH,07EH,07EH,000H
      00
7B26  18 3C 7E 18 7E 3C 18      DB      018H,03CH,07EH,018H,07EH,03CH,018H,0FFH
      FF
7B2E  18 3C 7E 18 18 18 18      DB      018H,03CH,07EH,018H,018H,018H,018H,000H
      00
7B36  18 18 18 18 7E 3C 18      DB      018H,018H,018H,018H,07EH,03CH,018H,000H
      00
7B3E  00 18 0C FE 0C 18 00      DB      000H,018H,00CH,0FEH,00CH,018H,000H,000H
      00
7B46  00 30 60 FE 60 30 00      DB      000H,030H,060H,0FEH,060H,030H,000H,000H
      00
7B4E  00 00 C0 C0 C0 FE 00      DB      000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H
      00
7B56  00 24 66 FF 66 24 00      DB      000H,024H,066H,0FFH,066H,024H,000H,000H
      00
7B5E  00 18 3C 7E FF FF 00      DB      000H,018H,03CH,07EH,0FFH,0FFH,000H,000H
      00
7B66  00 FF FF 7E 3C 18 00      DB      000H,0FFH,0FFH,07EH,03CH,018H,000H,000H
      00

7B6E  00 00 00 00 00 00 00      DB      000H,000H,000H,000H,000H,000H,000H,000H
      00
7B76  30 78 78 30 30 00 30      DB      030H,078H,078H,030H,030H,000H,030H,000H
      00
7B7E  6C 6C 6C 00 00 00 00      DB      06CH,06CH,06CH,000H,000H,000H,000H,000H
      00
7B86  6C 6C FE 6C FE 6C 6C      DB      06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H
      00
7B8E  30 7C C0 78 0C F8 30      DB      030H,07CH,0C0H,078H,00CH,0F8H,030H,000H
      00
7B96  00 C6 CC 18 30 66 C6      DB      000H,0C6H,0CCH,018H,030H,066H,0C6H,000H
      00
7B9E  38 6C 38 76 DC CC 76      DB      038H,06CH,038H,076H,0DCH,0CCH,076H,000H
      00
7BA6  60 60 C0 00 00 00 00      DB      060H,060H,0C0H,000H,000H,000H,000H,000H
      00
7BAE  18 30 60 60 60 30 18      DB      018H,030H,060H,060H,060H,030H,018H,000H
      00
7BB6  60 30 18 18 18 30 60      DB      060H,030H,018H,018H,018H,030H,060H,000H
      00
7BBE  00 66 3C FF 3C 66 00      DB      000H,066H,03CH,0FFH,03CH,066H,000H,000H
      00
7BC6  00 30 30 FC 30 30 00      DB      000H,030H,030H,0FCH,030H,030H,000H,000H
      00
7BCE  00 00 00 00 00 30 30      DB      000H,000H,000H,000H,000H,030H,030H,060H
      60
7BD6  00 00 00 FC 00 00 00      DB      000H,000H,000H,0FCH,000H,000H,000H,000H
      00
7BDE  00 00 00 00 00 30 30      DB      000H,000H,000H,000H,000H,030H,030H,000H
      00
7BE6  06 0C 18 30 60 C0 80      DB      006H,00CH,018H,030H,060H,0C0H,080H,000H
      00

7BEE  7C C6 CE DE F6 E6 7C      DB      07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H
      00
7BF6  30 70 30 30 30 30 FC      DB      030H,070H,030H,030H,030H,030H,0FCH,000H
```

```
                00
78FE  78 CC 0C 38 60 CC FC      DB      078H,0CCH,00CH,038H,060H,0CCH,0FCH,000H
                00
7C06  78 CC 0C 38 0C CC 78      DB      078H,0CCH,00CH,038H,00CH,0CCH,078H,000H
                00
7C0E  1C 3C 6C CC FE 0C 1E      DB      01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H
                00
7C16  FC C0 F8 0C 0C CC 78      DB      0FCH,0C0H,0F8H,00CH,00CH,0CCH,078H,000H
                00
7C1E  38 60 C0 F8 CC CC 78      DB      038H,060H,0C0H,0F8H,0CCH,0CCH,078H,000H
                00
7C26  FC CC 0C 18 30 30 30      DB      0FCH,0CCH,00CH,018H,030H,030H,030H,000H
                00
7C2E  78 CC CC 78 CC CC 78      DB      078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H
                00
7C36  78 CC CC 7C 0C 18 70      DB      078H,0CCH,0CCH,07CH,00CH,018H,070H,000H
                00
7C3E  00 30 30 00 00 30 30      DB      000H,030H,030H,000H,000H,030H,030H,000H
                00
7C46  00 30 30 00 00 30 30      DB      000H,030H,030H,000H,000H,030H,030H,060H
                60
7C4E  18 30 60 C0 60 30 18      DB      018H,030H,060H,0C0H,060H,030H,018H,000H
                00
7C56  00 00 FC 00 00 FC 00      DB      000H,000H,0FCH,000H,000H,0FCH,000H,000H
                00
7C5E  60 30 18 0C 18 30 60      DB      060H,030H,018H,00CH,018H,030H,060H,000H
                00
7C66  78 CC 0C 18 30 00 30      DB      078H,0CCH,00CH,018H,030H,000H,030H,000H
                00

7C6E  7C C6 DE DE DE C0 78      DB      07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H
                00
7C76  30 78 CC CC FC CC CC      DB      030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H
                00
7C7E  FC 66 66 7C 66 66 FC      DB      0FCH,066H,066H,07CH,066H,066H,0FCH,000H
                00
7C86  3C 66 C0 C0 C0 66 3C      DB      03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H
                00
7C8E  F8 6C 66 66 66 6C F8      DB      0F8H,06CH,066H,066H,066H,06CH,0F8H,000H
                00
7C96  FE 62 68 78 68 62 FE      DB      0FEH,062H,068H,078H,068H,062H,0FEH,000H
                00
7C9E  FE 62 68 78 68 60 F0      DB      0FEH,062H,068H,078H,068H,060H,0F0H,000H
                00
7CA6  3C 66 C0 C0 CE 66 3E      DB      03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H
                00
7CAE  CC CC CC FC CC CC CC      DB      0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H
                00
7CB6  78 30 30 30 30 30 78      DB      078H,030H,030H,030H,030H,030H,078H,000H
                00
7CBE  1E 0C 0C 0C CC CC 78      DB      01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H
                00
7CC6  E6 66 6C 78 6C 66 E6      DB      0E6H,066H,06CH,078H,06CH,066H,0E6H,000H
                00
7CCE  F0 60 60 60 62 66 FE      DB      0F0H,060H,060H,060H,062H,066H,0FEH,000H
                00
7CD6  C6 EE FE FE D6 C6 C6      DB      0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H
                00
7CDE  C6 E6 F6 DE CE C6 C6      DB      0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H
                00
7CE6  38 6C C6 C6 C6 6C 38      DB      038H,06CH,0C6H,0C6H,0C6H,06CH,038H,000H
                00

7CEE  FC 66 66 7C 60 60 F0      DB      0FCH,066H,066H,07CH,060H,060H,0F0H,000H
                00
7CF6  78 CC CC CC DC 78 1C      DB      078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H
                00
7CFE  FC 66 66 7C 6C 66 E6      DB      0FCH,066H,066H,07CH,06CH,066H,0E6H,000H
                00
7D06  78 CC E0 70 1C CC 78      DB      078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H
                00
7D0E  FC B4 30 30 30 30 78      DB      0FCH,0B4H,030H,030H,030H,030H,078H,000H
                00
7D16  CC CC CC CC CC CC FC      DB      0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H
                00
```

```
7D1E  CC CC CC CC CC 78 30 00          DB      0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H
7D26  C6 C6 C6 D6 FE EE C6 00          DB      0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H
7D2E  C6 C6 6C 38 38 6C C6 00          DB      0C6H,0C6H,06CH,038H,038H,06CH,0C6H,00H
7D36  CC CC CC 78 30 30 78 00          DB      0CCH,0CCH,0CCH,078H,030H,030H,078H,000H
7D3E  FE C6 8C 18 32 66 FE 00          DB      0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H
7D46  78 60 60 60 60 60 78 00          DB      078H,060H,060H,060H,060H,060H,078H,000H
7D4E  C0 60 30 18 0C 06 02 00          DB      0C0H,060H,030H,018H,00CH,006H,002H,000H
7D56  78 18 18 18 18 18 78 00          DB      078H,018H,018H,018H,018H,018H,078H,000H
7D5E  10 38 6C C6 00 00 00 00          DB      010H,038H,06CH,0C6H,000H,000H,000H,000H
7D66  00 00 00 00 00 00 00 FF          DB      000H,000H,000H,000H,000H,000H,000H,0FFH

7D6E  30 30 18 00 00 00 00 00          DB      030H,030H,018H,000H,000H,000H,000H,000H
7D76  00 00 78 0C 7C CC 76 00          DB      000H,000H,078H,00CH,07CH,0CCH,076H,000H
7D7E  E0 60 60 7C 66 66 DC 00          DB      0E0H,060H,060H,07CH,066H,066H,0DCH,000H
7D86  00 00 78 CC C0 CC 78 00          DB      000H,000H,078H,0CCH,0C0H,0CCH,078H,000H
7D8E  1C 0C 0C 7C CC CC 76 00          DB      01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H
7D96  00 00 78 CC FC C0 78 00          DB      000H,000H,078H,0CCH,0FCH,0C0H,078H,000H
7D9E  38 6C 60 F0 60 60 F0 00          DB      038H,06CH,060H,0F0H,060H,060H,0F0H,000H
7DA6  00 00 76 CC CC 7C 0C F8          DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,0F8H
7DAE  E0 60 6C 76 66 66 E6 00          DB      0E0H,060H,06CH,076H,066H,066H,0E6H,000H
7DB6  30 00 70 30 30 30 78 00          DB      030H,000H,070H,030H,030H,030H,078H,000H
7DBE  0C 00 0C 0C 0C CC CC 78          DB      00CH,000H,00CH,00CH,00CH,0CCH,0CCH,078H
7DC6  E0 60 66 6C 78 6C E6 00          DB      0E0H,060H,066H,06CH,078H,06CH,0E6H,000H
7DCE  70 30 30 30 30 30 78 00          DB      070H,030H,030H,030H,030H,030H,078H,000H
7DD6  00 00 CC FE FE D6 C6 00          DB      000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H
7DDE  00 00 F8 CC CC CC CC 00          DB      000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H
7DE6  00 00 78 CC CC CC 78 00          DB      000H,000H,078H,0CCH,0CCH,0CCH,078H,000H

7DEE  00 00 DC 66 66 7C 60 F0          DB      000H,000H,0DCH,066H,066H,07CH,060H,0F0H ;p
7DF6  00 00 76 CC CC 7C 0C 1E          DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,01EH ;q
7DFE  00 00 DC 76 66 60 F0 00          DB      000H,000H,0DCH,076H,066H,060H,0F0H,000H ;r
7E06  00 00 7C C0 78 0C F8 00          DB      000H,000H,07cH,0c0H,078H,00cH,0f8H,000H ;S

7E0E  10 30 7C 30 30 34 18 00          DB      010H,030H,07CH,030H,030H,034H,018H,000H ;T
7E16  00 00 CC CC CC CC 76 00          DB      000H,000H,0CCH,0CCH,0CCH,0CCH,076H,000H ;U
7E1E  00 00 CC CC CC 78 30 00          DB      000H,000H,0CCH,0CCH,0CCH,078H,030H,000H ;V
7E26  00 00 C6 D6 FE FE 6C 00          DB      000H,000H,0C6H,0D6H,0FEH,0FEH,06CH,000H ;W
7E2E  00 00 C6 6C 38 6C C6 00          DB      000H,000H,0C6H,06CH,038H,06CH,0C6H,000H ;X
7E36  00 00 CC CC CC 7C 0C F8          DB      000H,000H,0CCH,0CCH,0CCH,07CH,00CH,0F8H ;Y
7E3E  00 00 FC 98 30 64 FC             DB      000H,000H,0FCH,098H,030H,064H,0FCH,000H ;Z
```

```
      00
 7E46 1C 30 30 E0 30 30 1C            DB      01CH,030H,030H,0E0H,030H,030H,01CH,000H ;
      00
 7E4E 18 18 18 00 18 18 18            DB      018H,018H,018H,000H,018H,018H,018H,000H ;
      00
 7E56 E0 30 30 1C 30 30 E0            DB      0E0H,030H,030H,01CH,030H,030H,0E0H,000H ;
      00
 7E5E 76 DC 00 00 00 00 00            DB      076H,0DCH,000H,000H,000H,000H,000H,000H
      00
 7E66 00 10 38 6C C6 C6 FE            DB      000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H
      00


                                      SUBTTL  Reset Vector
                                ;*************************************************************************
                                ;* RESET                                                                 *
                                ;*       This is the FAR JUMP at the 8088s reset location (FFFF:0000)*
                                ;*                                                                       *
                                ;* Entry:    none                                                        *
                                ;* Exit:     none                                                        *
                                ;* Alters:   none                                                        *
                                ;* Calls:    none                                                        *
                                ;* Stack:    00H words                                                   *
                                ;*************************************************************************
                                      ASSUME  CS:NOTHING,DS:NOTHING,ES:NOTHING
 7E6E                           RESET   PROC    FAR
 7FF0                                   ORG     ROMSIZE * 1024 - 16
 7FF0 EA 0000 ---- R                    JMP     BIOS_START

 7FF6                                   ORG     ROMSIZE * 1024 - 10
 7FF6 30 39 2F 31 30 2F 38              DB      '09/10/86'
      36
 7FFE FE                                DB      0FEH
 7FFF 00                                DB      000H
                                RESET   ENDP
 8000                           CODE    ENDS
                                        END     BIOS_START
```

Macros:

Name	Lines
SCCIN	3
SCCOUT	3

Structures and Records:

Name	Width / Shift	# fields / Width	Mask	Initial
BOOTLOADER	003A	0006		
BL_START	0000			
BL_OEMNAME	0003			
BL_BPB	000B			
BL_NOTUSED	001E			
BL_IODIR	0024			
BL_MSDIR	002F			
BOOT_SECTOR	001E	0003		
START	0000			
OEM_NAME	0003			
DISKBPB	000B			
BPB	0013	000B		
BYTESPERSECTOR	0000			
SECTORSPERCLUSTER	0002			
RESERVEDSECTORS	0003			
NUMBEROFFATS	0005			
NUMBERDIRENTRIES	0006			
TOTALSECTORS	0008			
MEDIABYTE	000A			
SECTORSPERFAT	000B			
SECTORSPERTRACK	000D			
HEADS	000F			
HIDDENSECTORS	0011			
BUILDSTRUC	0016	0004		
MEDIAINFAT	000D			
SCRATCHADDRESS	000E			
BPBADDRESS	0012			
DAYTIME	0006	0005		
DAYS	0000			
MINUTES	0002			
HOURS	0003			
HUNDREDTHS	0004			
SECONDS	0005			
DEVATTRIB	0010	000D		
CHR	000F	0001	8000	0000
IOCTL	000E	0001	4000	0000
IBM	000D	0001	2000	0000
DMY	000C	0001	1000	0000
OPEN	000B	0001	0800	0000
DMY2	0007	0004	0780	0000
VER32	0006	0001	0040	0000
DMY3	0005	0001	0020	0000
SPC	0004	0001	0010	0000
CLK	0003	0001	0008	0000
NIL	0002	0001	0004	0000
STO	0001	0001	0002	0000
STI	0000	0001	0001	0000
DIRECTORYENTRY	0020	0009		
FILENAME	0000			
FILENAMEEXT	0008			
FILEATTRIBUTE	000B			
FILERESERVED	000C			
FILETIME	0016			
FILEDATE	0018			
FILECLUSTER1	001A			
FILESIZELOW	001C			
FILESIZEHIGH	001E			
FATID	0003	0003		
FATID1	0000			
FATID2	0001			
FATID3	0002			
GENIOCTLSTRC	0017	0006		
CATEGORY	000D			
IOCTLFUNCTION	000E			

```
  SICONTENTS . . . . . .          000F
  DICONTENTS . . . . . .          0011
  IOCTLPARAM . . . . . .          0013
INITSTRUC  . . . . . .            0017    0005
  NUMBEROFUNITS  . . . . .        000D
  BREAKADDRESS . . . . . .        000E
  BPBPOINTER . . . . . .          0012
  DEVICENUMBER . . . . . .        0016
IOCTLDEVPAR . . . . . .           002A    0007
  SFUNCTIONS . . . . . .          0000
  DEVICETYPE . . . . . .          0001
  DEVICEATTRIB . . . . . .        0002
  NUMCYLINDERS . . . . . .        0004
  MEDIATYPE  . . . . . .          0006
  CUR_BPB  . . . . . .            0007
  TRACKLAYOUT  . . . . . .        0008
LINKLIST . . . . . .              0012    0006
  NEXTDEVICE . . . . . .          0000
  NXTDEVSEG  . . . . . .          0002
  DEVATTRIBUTES  . . . . .        0004
  DEVSTRATEGY  . . . . . .        0006
  DEVINTERRUPT . . . . . .        0008
  DEVNAME  . . . . . .            000A
MCSTRUC  . . . . . .              0013    0004
  CURRENTMEDIA . . . . . .        000D
  MEDIASTATUS  . . . . . .        000E
  PREVOLUMEIDPTR . . . . . .      000F
RD_WR_FMT_TRK  . . . . . .        000D    0006
  SFUNCTION  . . . . . .          0000
  HEAD . . . . . .                0001
  CYLINDER . . . . . .            0003
  STARTSECTOR  . . . . . .        0005
  NUMOFSECTORS . . . . . .        0007
  TRANSFERADR  . . . . . .        0009
RNDSTRUC . . . . . .              000E    0002
  NEXTBYTE . . . . . .            000D
RWSTRUC  . . . . . .              001A    0006
  MEDIADESCRIPTOR  . . . . .      000D
  XFERADDRESS  . . . . . .        000E
  XFERCOUNT  . . . . . .          0012
  FIRSTSECTOR  . . . . . .        0014
  VOLUMEIDONERROR  . . . . .      0016
SERIAL_TABLE . . . . . .          000F    0009
  PORT_ADDRESS . . . . . .        0000
  BUFFER_SEG . . . . . .          0002
  BUFFER_START . . . . . .        0004
  BUFFER_END . . . . . .          0006
  BUFFER_HEAD  . . . . . .        0008
  BUFFER_TAIL  . . . . . .        000A
  TIMEOUT  . . . . . .            000C
  HANDSHAKE  . . . . . .          000D
  WR5COPY  . . . . . .            000E
STATICHDR  . . . . . .            000D    0005
  REQUESTLENGTH  . . . . .        0000
  UNIT . . . . . .                0001
  COMMAND  . . . . . .            0002
  STATUS . . . . . .              0003
  LINKS  . . . . . .              0005
STATREC  . . . . . .              0010    0005
  ERR  . . . . . .                000F    0001    8000    0000
  STATRES  . . . . . .            000A    0005    7C00    0000
  BUSY . . . . . .                0009    0001    0200    0000
  REQDONE  . . . . . .            0008    0001    0100    0000
  ERROR  . . . . . .              0000    0008    00FF    0000

Segments and Groups:

                N a m e         Size    Align   Combine Class

BOOT_CODE  . . . . . .          0000    AT      1000
CODE . . . . . .                8000    PARA    PUBLIC
DIAGNOSTICS  . . . . . .        0002    AT      D800
IODATA . . . . . .              06A8    AT      0000
OS_CODE  . . . . . .            0000    AT      0070
```

Symbols:

Name	Type	Value	Attr	
ACTIVE_PAGE	L BYTE	0462	IODATA	
ADDR_6845	L WORD	0463	IODATA	
ALT_INPUT	L BYTE	0419	IODATA	
B0	L NEAR	0060	CODE	
BAD_CMD	Number	0001		
BAD_LENGTH	Number	0005		
BASE_8250	Number	03F8		
BASIC	L DWORD	0060	IODATA	
BC1	L NEAR	1690	CODE	
BC2	L NEAR	1684	CODE	
BC3	L NEAR	168E	CODE	
BCDBYTE	N PROC	1011	CODE	Length = 0012
BCDTOHEX	N PROC	0FF4	CODE	Length = 001D
BC_RET	L NEAR	0355	CODE	
BE1	L NEAR	09DE	CODE	
BE2	L NEAR	0A0E	CODE	
BEEPER_PORT	Number	0061		
BELL	Number	0007		
BIOS	L NEAR	003C	CODE	
BIOS2	L NEAR	004F	CODE	
BIOSEND	Number	E000		
BIOSMSG	L BYTE	194F	CODE	
BIOSTART	Number	C800		
BIOS_BREAK	L BYTE	0471	IODATA	
BIOS_START	F PROC	0000	CODE	Length = 0103
BLANK	L BYTE	0A2C	CODE	
BLANK1	L BYTE	0A2D	CODE	
BLANK2	L BYTE	0A2E	CODE	
BOOT	F PROC	0908	CODE	Length = 0124
BOOTERR	L NEAR	0A2B	CODE	
BOOTOFFSET	Number	0006		
BOOT_LCN	L FAR	0000	BOOT_CODE	
BOOT_VECT	L DWORD	0064	IODATA	
BREAKPOINT	L DWORD	000C	IODATA	
BRKFLAG	L BYTE	0505	IODATA	
BRK_SVC	N PROC	1731	CODE	Length = 005F
BS1	L NEAR	1431	CODE	
BSV1	L NEAR	1766	CODE	
BSV2	L NEAR	1777	CODE	
BUF1	L BYTE	0552	IODATA	Length = 0020
BUF1_END	L BYTE	0572	IODATA	
BUF2	L BYTE	0572	IODATA	Length = 0020
BUF2_END	L BYTE	0592	IODATA	
BUF3	L BYTE	0592	IODATA	Length = 0020
BUF3_END	L BYTE	05B2	IODATA	
BUF4	L BYTE	05B2	IODATA	Length = 0020
BUF4_END	L BYTE	05D2	IODATA	
BUFFER_CHECK	N PROC	1672	CODE	Length = 001F
BUFFER_STATUS	L NEAR	1425	CODE	
BUFF_END	L WORD	0482	IODATA	
BUFF_START	L WORD	0480	IODATA	
BUFINIT	N PROC	123F	CODE	Length = 0084
BYTSPERSECT	L WORD	0913	CODE	
CB1	L NEAR	08E3	CODE	
CENTURY_DAYS	Number	1C89		
CHA14	Number	00B1		
CHANGEBOOT	N PROC	08C0	CODE	Length = 0048
CHAR_GEN_TBL	L BYTE	7A6E	CODE	
CHB14	Number	00B3		
CHC14	Number	00B5		
CHD14	Number	00B7		
CHECKSCC	N PROC	0542	CODE	Length = 004D
CHECK_ALL	L NEAR	1393	CODE	
CHECK_BUF	L NEAR	0310	CODE	
CHECK_COM1	Number	0003		
CHECK_RAM	N PROC	03F9	CODE	Length = 0017
CHK128K	L NEAR	088A	CODE	
CHKCONSOLE	N PROC	05B9	CODE	Length = 0016
CHKSCC	N PROC	0525	CODE	Length = 001D
CHKSCC3	L NEAR	0534	CODE	

CHK_BIOS	N PROC	0220	CODE	Length = 000A
CHK_ESC	N PROC	02E5	CODE	Length = 004F
CHK_EXTND	L NEAR	022E	CODE	
CHK_LEAP	N PROC	10CE	CODE	Length = 0010
CHK_RAM	L NEAR	0413	CODE	
CHK_VIDEO	N PROC	0208	CODE	Length = 0018
CHNGBOOT	L NEAR	08CC	CODE	
CLEARMSG	L BYTE	1A06	CODE	
CLEAR_RAM	N PROC	0452	CODE	Length = 0035
CLOCKINTERUPT	Number	001A		
CLOCK_INT	F PROC	0ECA	CODE	Length = 0074
CLOCK_TRAP	L DWORD	0068	IODATA	
CLR_CTS	N PROC	0E4A	CODE	Length = 0008
CLR_CTS?	N PROC	0E31	CODE	Length = 0019
CLR_RAM_RET	L NEAR	0485	CODE	
CLS_PTR	L WORD	06A6	IODATA	
CLS_STRING	L BYTE	1514	CODE	
CNTR0_INIT	Number	0036		
CNTR1_INIT	Number	0068		
CNTR2_INIT	Number	00B6		
CNVRT_DAYS	L NEAR	1071	CODE	
CNVRT_MONTH	L NEAR	105D	CODE	
CNVRT_TICKS	N PROC	0F47	CODE	Length = 004B
CNVRT_TIME	N PROC	0F92	CODE	Length = 0062
CNVRT_YEAR	L NEAR	104B	CODE	
COM1CHAR	L BYTE	05D2	IODATA	
COM1DATRDY	Number	0001		
COM1INBRK	L NEAR	0E19	CODE	
COM1INIT	L NEAR	0CC6	CODE	
COM1INRDY	L NEAR	0DE0	CODE	
COM1IN_INT	F PROC	0D9F	CODE	Length = 0092
COM1IN_TRAP	L DWORD	0030	IODATA	
COM1READY	L BYTE	05D3	IODATA	
COM1RECV	L NEAR	0D61	CODE	
COM1SEND	L NEAR	0D58	CODE	
COM1STAT	L NEAR	0CF5	CODE	
COM1TIMEOUT	Number	0080		
COM1TOCON	N PROC	05CF	CODE	Length = 001A
COM1WRITE	L NEAR	0D32	CODE	
COM1XMITRDY	Number	0020		
COM1_DEV_NUM	Number	0000		
COM1_INT	F PROC	0C95	CODE	Length = 00ED
COM1_TIMEOUT	L NEAR	0D4A	CODE	
COM1_TRAP	Number	0014		
COM1_VECT	L DWORD	0050	IODATA	
CONINOUT_TRAP	L DWORD	01C8	IODATA	
CONTHEXTOBCD	L NEAR	010E	CODE	
CONTINUE	L NEAR	0255	CODE	
CONVERT_DATE	N PROC	102F	CODE	Length = 004B
CONVERT_DAYS	N PROC	107A	CODE	Length = 0054
CON_DEV_NUM	Number	0000		
CON_RESET_ENABLE	L BYTE	05D8	IODATA	
COPRID	Number	0062		
COPRPORT	Number	0062		
CR	Number	000D		
CRC_ERROR	Number	0004		
CRLF	L BYTE	18CD	CODE	
CRT_COLS	L WORD	044A	IODATA	
CRT_LENGTH	L WORD	044C	IODATA	
CRT_MODE	L BYTE	0449	IODATA	
CRT_MODE_SET	L BYTE	0465	IODATA	
CRT_PALETTE	L BYTE	0466	IODATA	
CRT_START	L WORD	044E	IODATA	
CTLC	Number	0003		
CTLP	Number	0010		
CTL_BREAK	Number	0000		
CTL_PRNSCR	Number	0072		
CURS_MODE	L WORD	0460	IODATA	
CURS_POSITION	L WORD	0450	IODATA	Length = 0008
CVT_MONTH	L NEAR	1061	CODE	
DAYSCOUNTED	L WORD	0514	IODATA	
DAYSPERMONTH	L BYTE	1023	CODE	
DDS	N PROC	03F1	CODE	Length = 0008
DEFAULT_BUFF	Number	000A		
DELAYMESG	N PROC	013B	CODE	Length = 0012

```
DELAYMSG . . . . . .  L BYTE  19E6  CODE
DELAY_LOOP . . . . . .  L NEAR  0133  CODE
DELMSGOUT  . . . . . .  L NEAR  0149  CODE
DIAG . . . . . .  F PROC  0002  DIAGNOSTICS  Length = 0000
DIAG_ID  . . . . . .  Number  5A5A
DIAG_ID_LOC  . . . . . .  L WORD  0000  DIAGNOSTICS
DISKERROR  . . . . . .  L BYTE  181C  CODE
DISKMASK . . . . . .  L NEAR  0857  CODE
DISKNOTB . . . . . .  L NEAR  0393  CODE
DISKSEG  . . . . . .  Number  E000
DISK_DATA_SEG  . . . . . .  L WORD  048D  IODATA
DISK_PTR . . . . . .  L DWORD 0078  IODATA
DISK_STATUS  . . . . . .  L BYTE  0441  IODATA
DISK_TRK . . . . . .  L BYTE  0495  IODATA
DIV110 . . . . . .  Number  0417
DIV75  . . . . . .  Number  0600
DIVIDE_ERR . . . . . .  L DWORD 0000  IODATA
DOBOOT . . . . . .  L NEAR  00F7  CODE
DOPOST . . . . . .  L NEAR  0103  CODE
DRVCOUNT . . . . . .  L BYTE  06A1  IODATA
DRVSTACK . . . . . .  L WORD  0659  IODATA
DRV_TYPE_TBL . . . . . .  L BYTE  0492  IODATA
DSR_ACTIVE . . . . . .  L NEAR  0D12  CODE
DSR_READY  . . . . . .  L NEAR  0D50  CODE
DUMMY_IRET . . . . . .  L NEAR  06FA  CODE

E14A_TRAP  . . . . . .  L DWORD 0228  IODATA
E14B_TRAP  . . . . . .  L DWORD 020B  IODATA
E14C_TRAP  . . . . . .  L DWORD 022C  IODATA
E14D_TRAP  . . . . . .  L DWORD 020C  IODATA
E63A_TRAP  . . . . . .  L DWORD 0268  IODATA
E63B_TRAP  . . . . . .  L DWORD 0248  IODATA
ENABLE_SCC . . . . . .  N PROC  058F  CODE  Length = 002A
EQUIP_FLAG . . . . . .  L WORD  0410  IODATA
EQUIP_VECT . . . . . .  L DWORD 0044  IODATA
ERROR_STOP . . . . . .  L NEAR  0101  CODE
ERRRET . . . . . .  L NEAR  0357  CODE
ERR_RETURN . . . . . .  L NEAR  1328  CODE
ESC  . . . . . .  Number  001B
ESC_FLAG . . . . . .  L BYTE  06A5  IODATA
ETX  . . . . . .  Number  0003
EVENKBYTE  . . . . . .  L NEAR  063D  CODE
EXITINITSCC  . . . . . .  L NEAR  050A  CODE
EXIT_PRTSCR  . . . . . .  L NEAR  1AD4  CODE
EXTND_BIOS . . . . . .  N PROC  022A  CODE  Length = 003C
EXTND_BIOS_ID  . . . . . .  Number  AA55

FACTRFIVE  . . . . . .  L WORD  0F3E  CODE
FACTRHUNS  . . . . . .  L WORD  0F42  CODE
FACTRMINS  . . . . . .  L BYTE  0F46  CODE
FACTRSECS  . . . . . .  L WORD  0F44  CODE
FALSE  . . . . . .  Number  0000
FATAL_ERROR  . . . . . .  L NEAR  00FA  CODE
FFPORT . . . . . .  Number  00FF
FIND_DISK  . . . . . .  N PROC  0432  CODE  Length = 0020
FIND_RAM_TOP . . . . . .  N PROC  0410  CODE  Length = 0022
FLOPPYBOOT . . . . . .  L NEAR  029F  CODE
FLOPPYFLAG . . . . . .  Number  0010
FLOPPYRETURN . . . . . .  L NEAR  028F  CODE
FLOPPYSEG  . . . . . .  Number  CC00
FORMAT_RAM . . . . . .  N PROC  0B61  CODE  Length = 005F
FORMAT_TRK_CODE  . . . . . .  Number  0042
FOUND_DISK . . . . . .  L NEAR  044D  CODE
FR1  . . . . . .  L NEAR  0890  CODE
FR2  . . . . . .  L NEAR  0871  CODE
FR4  . . . . . .  L NEAR  0887  CODE
FR5  . . . . . .  L NEAR  0888  CODE

GAMEPORT . . . . . .  Number  0201
GENERAL_FAILURE  . . . . . .  Number  000C
GETEQUIP . . . . . .  F PROC  064D  CODE  Length = 000A
GETMEMSIZE . . . . . .  F PROC  0643  CODE  Length = 000A
GET_DEV_CODE . . . . . .  Number  0060

HARD_DISK  . . . . . .  Number  0005
HEADMORE . . . . . .  L NEAR  0E40  CODE
```

```
HEXBYTE . . . . . L NEAR  10FB  CODE
HEXTOBCD . . . . . N PROC  01C9  CODE    Length = 003F
HEXTOBCDLOOP . . . . . L NEAR  01D0  CODE
HEX_DWORD . . . . . N PROC  10DE  CODE    Length = 0045
HEX_OUT_BYTE . . . . . N PROC  049B  CODE    Length = 0017
HEX_OUT_NIBL . . . . . N PROC  04AF  CODE    Length = 001E
HEX_OUT_WORD . . . . . N PROC  04CD  CODE    Length = 000D
HEX_TABLE . . . . . L NEAR  048D  CODE
HF_CNTL . . . . . L BYTE  0490  IODATA
HS1 . . . . . L NEAR  1449  CODE
HS2 . . . . . L NEAR  1458  CODE
HS3 . . . . . L NEAR  1466  CODE
HS4 . . . . . L NEAR  147C  CODE
HS_CNTRL . . . . . L NEAR  143A  CODE
H_INTR_MSG . . . . . L BYTE  190B  CODE

ID1 . . . . . L NEAR  0832  CODE
IDENTIFY . . . . . L NEAR  0818  CODE
IDSIZE . . . . . Number  0008
IN1 . . . . . L NEAR  1336  CODE
IN2 . . . . . L NEAR  134A  CODE
IN3 . . . . . L NEAR  1357  CODE
INIT2SCC . . . . . N PROC  1123  CODE    Length = 0066
INIT4SCC . . . . . L NEAR  0510  CODE
INITCOMS . . . . . N PROC  072D  CODE    Length = 004C
INITCOMSLOOP . . . . . L NEAR  0738  CODE
INITIALIZE . . . . . L NEAR  1332  CODE
INITMORSCC . . . . . N PROC  1189  CODE    Length = 0054
INITPRNS . . . . . N PROC  077F  CODE    Length = 0045
INITPRNSLOOP . . . . . L NEAR  078A  CODE
INITRAMDISK . . . . . N PROC  0800  CODE    Length = 0061
INITSCC . . . . . N PROC  11DD  CODE    Length = 0062
INIT_8254 . . . . . N PROC  0710  CODE    Length = 001D
INIT_8259 . . . . . N PROC  06FB  CODE    Length = 0015
INIT_COM1 . . . . . Number  0000
INIT_DELAY . . . . . L NEAR  012D  CODE
INIT_FLAG . . . . . L BYTE  0412  IODATA
INIT_HDWR . . . . . N PROC  0657  CODE    Length = 0018
INIT_MEMORY . . . . . L NEAR  0458  CODE
INIT_MEMSIZE . . . . . N PROC  062C  CODE    Length = 0017
INNULSEQ . . . . . L BYTE  05D4  IODATA
INSIDE . . . . . L NEAR  0E8B  CODE
INTEND . . . . . Number  0020
INTERUPT_MASK . . . . . Number  00FE
INTER_FLAG . . . . . L BYTE  046B  IODATA
INTMASK_8250 . . . . . Number  00EF
INTR_ERR . . . . . F PROC  17C5  CODE    Length = 0039
INTR_ERR_RET . . . . . L NEAR  17F9  CODE
INTR_HARD . . . . . L NEAR  17F2  CODE
INTR_SOFT . . . . . L NEAR  17DE  CODE
INTR_SP . . . . . L WORD  069D  IODATA
INTR_SS . . . . . L WORD  069F  IODATA
INTSTACK . . . . . L WORD  0699  IODATA
INVALID_DISK_CHANGE . . . . . Number  000F
IODATAEND . . . . . L BYTE  06A8  IODATA
IOSYSDIR . . . . . L BYTE  17FE  CODE
IOSYS_NAME . . . . . L BYTE  0926  CODE
IO_ROM_INIT . . . . . L WORD  0467  IODATA
IO_ROM_SEG . . . . . L WORD  0469  IODATA
IRET_INST . . . . . L NEAR  17C4  CODE
ISTESTEDMESG . . . . . L BYTE  1902  CODE
ISTESTEDMSG . . . . . N PROC  01AD  CODE    Length = 0008

KBD_BREAK . . . . . L DWORD 006C  IODATA
KBD_BUFFER . . . . . L WORD  041E  IODATA  Length = 0010
KBD_BUF_END . . . . . L WORD  043E  IODATA
KBD_BUF_FULL . . . . . L NEAR  0E2B  CODE
KBD_BUF_HEAD . . . . . L WORD  041A  IODATA
KBD_BUF_TAIL . . . . . L WORD  041C  IODATA
KBD_FLAG . . . . . L BYTE  0417  IODATA
KBD_FLAG_1 . . . . . L BYTE  0418  IODATA
KBD_FLAG_2 . . . . . L BYTE  0498  IODATA
KYBD_CHECK . . . . . Number  0001
KYBD_PORT . . . . . Number  0061
KYBD_READ . . . . . Number  0000
KYBD_TRAP . . . . . Number  0016
```

```
KYBD_VECT . . . . . .  L DWORD 0058   IODATA

LASTSEG . . . . . .  L NEAR  046E   CODE
LB1 . . . . . .  L NEAR  02C0   CODE
LB2 . . . . . .  L NEAR  02D4   CODE
LB_FAIL . . . . . .  L NEAR  02E4   CODE
LB_SUCCESS . . . . . .  L NEAR  02DD   CODE
LEAPMONTH . . . . . .  L BYTE  1024   CODE
LF . . . . . .  Number  000A
LOADBOOTSECT . . . . . .  N PROC  0285   CODE    Length = 0060
LOAD_OS . . . . . .  L NEAR  0958   CODE
LOOP1 . . . . . .  L NEAR  10FF   CODE
LPT1_DEV_NUM . . . . . .  Number  0000
LPT2_DEV_NUM . . . . . .  Number  0001
LPT3_DEV_NUM . . . . . .  Number  0002
LPT_INIT . . . . . .  Number  0001
LPT_SEND . . . . . .  Number  0000
LPT_STAT . . . . . .  Number  0002
LPT_TRAP . . . . . .  Number  0017

MAXHOURSLS . . . . . .  Number  00B0
MAXHOURSMS . . . . . .  Number  0018
MAX_COMMAND . . . . . .  Number  0011
MDISKINT . . . . . .  L DWORD 01DC   IODATA
MDISK_TRAP . . . . . .  Number  0077
MEMDISKID . . . . . .  L BYTE  1814   CODE
MEMDISK_IO . . . . . .  F PROC  0334   CODE    Length = 00BD
MEMDSKBOOT . . . . . .  L NEAR  0287   CODE
MEMDSKMASK . . . . . .  L WORD  06A3   IODATA
MEMSIZE . . . . . .  L NEAR  0168   CODE
MEMSIZE_VECT . . . . . .  L DWORD 0048   IODATA
MEMTEST . . . . . .  L WORD  03FC   IODATA
MEMTESTLOOP . . . . . .  L NEAR  0156   CODE
MEMTESTMESG . . . . . .  L BYTE  1995   CODE
MEMTESTMSG . . . . . .  N PROC  0196   CODE    Length = 0008
MEM_SIZE . . . . . .  L WORD  0413   IODATA
MESSAGE . . . . . .  N PROC  0487   CODE    Length = 0011
MESSAGE_RET . . . . . .  L NEAR  0494   CODE
MFG_ERR_FLAG . . . . . .  L BYTE  0415   IODATA
MIO . . . . . .  L NEAR  0334   CODE
MONTHLOOP . . . . . .  L NEAR  10A8   CODE
MOTOR_COUNT . . . . . .  L BYTE  0440   IODATA
MOTOR_STATUS . . . . . .  L BYTE  043F   IODATA
MOVEADDR . . . . . .  N PROC  12C3   CODE    Length = 0008
MRET . . . . . .  L NEAR  035E   CODE
MSDOSDIR . . . . . .  L BYTE  1809   CODE
MSGEND . . . . . .  Number  0024
MSSYS_NAME . . . . . .  L BYTE  0931   CODE

NEC_STATUS . . . . . .  L BYTE  0442   IODATA  Length = 0007
NEXT_CHAR . . . . . .  L NEAR  048A   CODE
NMI . . . . . .  L DWORD 0008   IODATA
NOCOM . . . . . .  L NEAR  0758   CODE
NOCOM1_RESET . . . . . .  L NEAR  0DFE   CODE
NOGAMES . . . . . .  L NEAR  070E   CODE
NOHEADS . . . . . .  L WORD  0922   CODE
NOLSOFLW . . . . . .  L NEAR  0C4C   CODE
NOPRN . . . . . .  L NEAR  07A2   CODE
NOTLEAPMONTH . . . . . .  L NEAR  10C1   CODE
NOTLEAPYEAR . . . . . .  L NEAR  1098   CODE
NOTMOTTIME . . . . . .  L NEAR  0C83   CODE
NOTRAMDSKB . . . . . .  L NEAR  0842   CODE
NOT_EGA_KBD . . . . . .  L NEAR  0300   CODE
NOT_KBD_END . . . . . .  L NEAR  0E08   CODE
NOT_LEAP . . . . . .  L NEAR  1059   CODE
NOT_READY . . . . . .  Number  0002
NO_RESET . . . . . .  L NEAR  1607   CODE
NO_SEND . . . . . .  L NEAR  138E   CODE
NO_VOLUME . . . . . .  L NEAR  097C   CODE
NUL . . . . . .  Number  0000
NXT_EXT_BIO . . . . . .  L NEAR  025B   CODE

OK1 . . . . . .  L NEAR  098F   CODE
OK2 . . . . . .  L NEAR  09BF   CODE
OK3 . . . . . .  L NEAR  09E0   CODE
OK4 . . . . . .  L NEAR  0A15   CODE
```

OK5	L NEAR	0A10	CODE	
OKRET	L NEAR	035B	CODE	
ONESOF	L NEAR	01E1	CODE	
OS_BASE	L FAR	0000	OS_CODE	
OUTSIDE	L NEAR	0E89	CODE	
OUT_OF_PAPER	Number	0009		
OVERFLOW	L DWORD	0010	IODATA	
PARITYMSG	L BYTE	18BD	CODE	
PARITY_ERROR	F PROC	17AB	CODE	Length = 001A
PE0	L NEAR	17B2	CODE	
PE_RET	L NEAR	17C1	CODE	
PHYS_FMT_FLAG	L BYTE	048F	IODATA	
PORT0_8254	Number	0040		
PORT1_8254	Number	0041		
PORT2_8254	Number	0042		
PORT7712	Number	0039		
PORTCNTL_8254	Number	0043		
PORTCNTL_8259	Number	0020		
PORTCNTL_FDC	Number	02D8		
PORTDATA_8259	Number	0021		
PORT_STATUS	L NEAR	1360	CODE	
POST	N PROC	014D	CODE	Length = 0020
PRINTER_BASE	L WORD	0408	IODATA	Length = 0004
PRINTINIT	L NEAR	0EBB	CODE	
PRINTOUT	L NEAR	0E85	CODE	
PRINTSTAT	L NEAR	0EB0	CODE	
PRINT_AL	N PROC	1AF3	CODE	Length = 000D
PRINT_CHAR	N PROC	1AE1	CODE	Length = 0012
PRINT_INT	F PROC	0E55	CODE	Length = 0075
PRINT_SCREEN	F PROC	1A80	CODE	Length = 0061
PRINT_VECT	L DWORD	005C	IODATA	
PRNT_CHAR	L NEAR	1AEF	CODE	
PRNT_CRLF	N PROC	1B00	CODE	Length = 000D
PRNT_ERROR	L NEAR	1ADA	CODE	
PRNT_SCRN	L DWORD	0014	IODATA	
PRNT_TIMEOUT	L BYTE	0478	IODATA	Length = 0004
PRN_NXT_CHR	L NEAR	1AB2	CODE	
PS1	L NEAR	1381	CODE	
PTRSAVE	L DWORD	0510	IODATA	
R14A_TRAP	L DWORD	0230	IODATA	
R14B_TRAP	L DWORD	0210	IODATA	
R14C_TRAP	L DWORD	0234	IODATA	
R14D_TRAP	L DWORD	0214	IODATA	
R2	L NEAR	13EE	CODE	
R3	L NEAR	1422	CODE	
R63A_TRAP	L DWORD	0270	IODATA	
R63B_TRAP	L DWORD	0250	IODATA	
RAMBASEPORT	Number	0284		
RAMBOOTSECT	L NEAR	0908	CODE	
RAMDIRENTRY	L WORD	0919	CODE	
RAMDRVBFLG	L BYTE	06A2	IODATA	
RAMFATSECT	L WORD	091E	CODE	
RAMRSVSECT	L WORD	0916	CODE	
RAMTOTALSECT	L WORD	091B	CODE	
RCV_SVC	N PROC	15B9	CODE	Length = 00B9
RD_SECTOR	Number	0000		
READ	L NEAR	1305	CODE	
READ_DATE	L NEAR	0F2C	CODE	
READ_FAULT	Number	0008		
READ_TICK	L NEAR	0EE8	CODE	
READ_TIME	L NEAR	0F0E	CODE	
READ_TRK_CODE	Number	0061		
RECAL_MAP	L BYTE	0491	IODATA	
RECV_COM1	Number	0002		
REINIT_8250	N PROC	05E9	CODE	Length = 002C
REINIT_8259	N PROC	0615	CODE	Length = 0017
RESET	F PROC	7E6E	CODE	Length = 0192
RESET_CNTRL	L NEAR	1493	CODE	
RESET_FLAG	L WORD	0472	IODATA	
RES_BIOS_FLAG	L WORD	04BB	IODATA	
RETURN	L NEAR	132A	CODE	
ROMBASEPORT	Number	0280		
ROMSIZE	Number	0020		
RS232_BASE	L WORD	0400	IODATA	Length = 0004

```
RS232_TIMEOUT . . . . . .  L BYTE  047C  IODATA  Length = 0004
RST_CLS_PTR . . . . . .  L NEAR  1504  CODE
RTO . . . . . .  L NEAR  15D6  CODE
RT1 . . . . . .  L NEAR  165C  CODE
RT2 . . . . . .  L NEAR  1610  CODE
RT3 . . . . . .  L NEAR  1621  CODE
RT4 . . . . . .  L NEAR  1625  CODE
RT5 . . . . . .  L NEAR  1656  CODE
RTC_HICOUNT . . . . . .  L WORD  049F  IODATA
RTC_LOWCOUNT . . . . . .  L WORD  049D  IODATA
RTC_USER_ADDR . . . . . .  L WORD  0499  IODATA
RTC_USER_SEG . . . . . .  L WORD  049B  IODATA
RTC_WAIT . . . . . .  L BYTE  04A1  IODATA
RUTNE_ESBX . . . . . .  N PROC  0266  CODE  Length = 001F

S14A_TRAP . . . . . .  L DWORD 0238  IODATA
S14B_TRAP . . . . . .  L DWORD 0218  IODATA
S14C_TRAP . . . . . .  L DWORD 023C  IODATA
S14D_TRAP . . . . . .  L DWORD 021C  IODATA
S63A_TRAP . . . . . .  L DWORD 0278  IODATA
S63B_TRAP . . . . . .  L DWORD 0258  IODATA
SAVE_CLS_PTR . . . . . .  L NEAR  1507  CODE
SB1 . . . . . .  L NEAR  12E9  CODE
SB2 . . . . . .  L NEAR  12E2  CODE
SCANCODE . . . . . .  L BYTE  05D6  IODATA
SCC2 . . . . . .  L NEAR  050F  CODE
SCCBIOS . . . . . .  F PROC  12CB  CODE  Length = 01CE
SCCFLAG . . . . . .  L BYTE  05D7  IODATA
SCCINIT . . . . . .  N PROC  04DA  CODE  Length = 004B
SCC_BUF_READ . . . . . .  Number  0002
SCC_BUF_STAT . . . . . .  Number  0004
SCC_FLUSH . . . . . .  N PROC  1790  CODE  Length = 001B
SCC_INIT . . . . . .  Number  0000
SCC_NOT_FULL . . . . . .  L NEAR  161B  CODE
SCC_PORT_STAT . . . . . .  Number  0003
SCC_SEND . . . . . .  Number  0001
SCC_TRAP . . . . . .  Number  0071
SCSIBOOT . . . . . .  L NEAR  02AC  CODE
SCSIDELAY . . . . . .  N PROC  0126  CODE  Length = 0015
SCSIFLAG . . . . . .  Number  0004
SCSISEG . . . . . .  Number  C800
SCSI_INT_ADR . . . . . .  L WORD  0474  IODATA
SCSI_INT_SEG . . . . . .  L WORD  0476  IODATA
SECTOR_NOT_FOUND . . . . . .  Number  0008
SECTTRACK . . . . . .  L WORD  0920  CODE
SEEK_ERROR . . . . . .  Number  0006
SEEK_STATUS . . . . . .  L BYTE  043E  IODATA
SEND . . . . . .  L NEAR  1385  CODE
SER1TABLE . . . . . .  L WORD  1A26  CODE
SER1_BRKINT . . . . . .  F PROC  1691  CODE  Length = 0028
SER1_DATA . . . . . .  L BYTE  0516  IODATA  Length = 000F
SER1_DEV_NUM . . . . . .  Number  0000
SER1_INT . . . . . .  F PROC  1519  CODE  Length = 0028
SER2TABLE . . . . . .  L WORD  1A44  CODE
SER2_BRKINT . . . . . .  F PROC  16B9  CODE  Length = 0028
SER2_DATA . . . . . .  L BYTE  0525  IODATA  Length = 000F
SER2_DEV_NUM . . . . . .  Number  0001
SER2_INT . . . . . .  F PROC  1541  CODE  Length = 0028
SER3TABLE . . . . . .  L WORD  1A62  CODE
SER3_BRKINT . . . . . .  F PROC  16E1  CODE  Length = 0028
SER3_DATA . . . . . .  L BYTE  0534  IODATA  Length = 000F
SER3_DEV_NUM . . . . . .  Number  0002
SER3_INT . . . . . .  F PROC  1569  CODE  Length = 0028
SER4TABLE . . . . . .  L WORD  1A62  CODE
SER4_BRKINT . . . . . .  F PROC  1709  CODE  Length = 0028
SER4_DATA . . . . . .  L BYTE  0543  IODATA  Length = 000F
SER4_DEV_NUM . . . . . .  Number  0003
SER4_INT . . . . . .  F PROC  1591  CODE  Length = 0028
SETCOM1RATE . . . . . .  L NEAR  0CE1  CODE
SETDAYS . . . . . .  L NEAR  10C6  CODE
SETDSKMASK . . . . . .  L NEAR  082F  CODE
SETEQUIP . . . . . .  L NEAR  07FB  CODE
SETFLGXIT . . . . . .  L NEAR  0E9E  CODE
SETMONTH . . . . . .  L NEAR  10A2  CODE
SET_BUFFER . . . . . .  L NEAR  147F  CODE
SET_DATE . . . . . .  L NEAR  0F35  CODE
```

```
SET_DEV_CODE . . . . . .  Number  0040
SET_EQUIP  . . . . . .    N PROC  07CC  CODE    Length = 0034
SET_TIME . . . . . .      L NEAR  0F24  CODE
SFL1 . . . . . .          L NEAR  17AA  CODE
SFL2 . . . . . .          L NEAR  1790  CODE
SHOWKBYTES . . . . . .    N PROC  0185  CODE    Length = 0014
SHOWMEMSIZE  . . . . . .  N PROC  019E  CODE    Length = 000F
SINGLE_STEP  . . . . . .  L DWORD 0004  IODATA
SKIP_POST  . . . . . .    L NEAR  00E6  CODE
SPLRSVSECT . . . . . .    L WORD  0924  CODE
STACK_SP . . . . . .      L WORD  0699  IODATA
STACK_SS . . . . . .      L WORD  069B  IODATA
STATUS_BYTE  . . . . . .  L BYTE  0500  IODATA
STROBE . . . . . .        L NEAR  0EA5  CODE
STROBEFINI . . . . . .    L NEAR  0EA9  CODE
SYSERROR . . . . . .      L BYTE  1863  CODE
S_INTR_MSG . . . . . .    L BYTE  18D0  CODE
S_INTR_MSG2  . . . . . .  L BYTE  18F6  CODE

TABLE  . . . . . .        L BYTE  0587  CODE
TBL_COMPORT  . . . . . .  L WORD  0779  CODE
TBL_PRNPORT  . . . . . .  L WORD  07C4  CODE
TEST1  . . . . . .        L NEAR  12F7  CODE
TEST2  . . . . . .        L NEAR  12FE  CODE
TEST3  . . . . . .        L NEAR  1305  CODE
TEST4  . . . . . .        L NEAR  130C  CODE
TEST5  . . . . . .        L NEAR  1313  CODE
TEST6  . . . . . .        L NEAR  131A  CODE
TEST7  . . . . . .        L NEAR  1321  CODE
TESTBLKLOOP  . . . . . .  L NEAR  0176  CODE
TESTBLKXIT . . . . . .    L NEAR  0193  CODE
TESTBLOCK  . . . . . .    N PROC  016D  CODE    Length = 0029
TESTEDMSG  . . . . . .    L BYTE  1988  CODE
THISCENTURY  . . . . . .  L NEAR  1088  CODE
TICKSPERHUN  . . . . . .  L WORD  0F40  CODE
TICK_INT . . . . . .      F PROC  0C2C  CODE    Length = 0069
TICK_TRAP  . . . . . .    L DWORD 0020  IODATA
TIMER_BREAK  . . . . . .  L DWORD 0070  IODATA
TIMER_HIGH . . . . . .    L WORD  046E  IODATA
TIMER_LOW  . . . . . .    L WORD  046C  IODATA
TIMER_OFLW . . . . . .    L BYTE  0470  IODATA
TIMER_TRAP . . . . . .    L DWORD 01CC  IODATA
TIME_OUT . . . . . .      L NEAR  13CF  CODE
TRUE . . . . . .          Number  FFFF
TTY_TRAP . . . . . .      Number  0029
TTY_VECT . . . . . .      L DWORD 00A4  IODATA
TURNDISKOFF  . . . . . .  L NEAR  0C78  CODE

UNKNOWN_COMMAND  . . . . . .  Number  0003
UNKNOWN_MEDIA  . . . . . .    Number  0007
UNKNOWN_UNIT . . . . . .      Number  0001
USERINT  . . . . . .          Number  001C

VAL_GET_DATE . . . . . .  Number  0004
VAL_GET_TICK . . . . . .  Number  0000
VAL_GET_TIME . . . . . .  Number  0002
VAL_SET_DATE . . . . . .  Number  0005
VAL_SET_TICK . . . . . .  Number  0001
VAL_SET_TIME . . . . . .  Number  0003
VECT14A  . . . . . .      Number  0080
VECT14C  . . . . . .      Number  0081
VECTOR_INIT  . . . . . .  N PROC  0672  CODE    Length = 0089
VECTSCC_TRAP . . . . . .  L DWORD 01C4  IODATA
VER_TRK_CODE . . . . . .  Number  0062
VIDEO  . . . . . .        L DWORD 0040  IODATA
VIDEOEND . . . . . .      Number  C800
VIDEOSTART . . . . . .    Number  C000
VIDEO_GRAPH  . . . . . .  L DWORD 007C  IODATA
VIDEO_INIT . . . . . .    L DWORD 0074  IODATA
VIDEO_INT  . . . . . .    Number  0010
VIDEO_INT42  . . . . . .  L DWORD 0108  IODATA
VIDEO_MODE . . . . . .    Number  0000
VIDEO_PARAMS . . . . . .  L BYTE  70A4  CODE
VIDEO_STATE  . . . . . .  Number  000F
VIDEO_TTY  . . . . . .    Number  000E
```

```
WAITINIT . . . . . .  L NEAR  0EC5  CODE
WAIT_FOR_STATUS  . . . . . .  N PROC  1499  CODE  Length = 001D
WAIT_LOOP  . . . . . .  L NEAR  0D8B  CODE
WAIT_READY . . . . . .  N PROC  0D82  CODE  Length = 001D
WFS0 . . . . . .  L NEAR  149D  CODE
WFS1 . . . . . .  L NEAR  149F  CODE
WFS_END  . . . . . .  L NEAR  1484  CODE
WRITE_FAIL . . . . . .  Number  0003
WRITE_FAULT  . . . . . .  Number  000A
WRITE_PROTECT  . . . . . .  Number  0000
WRITE_TICK . . . . . .  L NEAR  0EFE  CODE
WRITE_TRK_CODE . . . . . .  Number  0041
WRITE_TTY  . . . . . .  F PROC  14B6  CODE  Length = 0063
WRPRNTERR  . . . . . .  L NEAR  0E81  CODE
WRTO@@CON  . . . . . .  L NEAR  14DB  CODE
WRTOVIDEO  . . . . . .  L NEAR  14E4  CODE
WR_SECTOR  . . . . . .  Number  0001

X1 . . . . . .  L NEAR  03A6  CODE
X2 . . . . . .  L NEAR  03AA  CODE
X3 . . . . . .  L NEAR  03D7  CODE
X6 . . . . . .  L NEAR  03DC  CODE
X638_TRAP  . . . . . .  L DWORD 0240  IDATA
X7 . . . . . .  L NEAR  03EE  CODE
XFER . . . . . .  L NEAR  0368  CODE
XIT2SCC  . . . . . .  L NEAR  0512  CODE
XITCHKCONSOLE  . . . . . .  L NEAR  05CE  CODE
XITCHKSCC  . . . . . .  L NEAR  0540  CODE
XITCHK_ESC . . . . . .  L NEAR  0330  CODE
XITCLOCKINT  . . . . . .  L NEAR  0EE8  CODE
XITCLRCTS  . . . . . .  L NEAR  0E49  CODE
XITCOM1  . . . . . .  L NEAR  0CC0  CODE
XITCOM1ERR . . . . . .  L NEAR  0CBE  CODE
XITCOM1IN  . . . . . .  L NEAR  0DCE  CODE
XITEQUIP . . . . . .  L NEAR  07EF  CODE
XITHEXBYTE . . . . . .  L NEAR  1112  CODE
XITHEXTOBCD  . . . . . .  L NEAR  0203  CODE
XITINISCC  . . . . . .  L NEAR  123E  CODE
XITINITCOMS  . . . . . .  L NEAR  075F  CODE
XITINITPRNS  . . . . . .  L NEAR  07A8  CODE
XITPRINTINT  . . . . . .  L NEAR  0E7B  CODE
XITRAMSIZE . . . . . .  L NEAR  042E  CODE
XITSCCINIT . . . . . .  L NEAR  04F4  CODE
XITWAITRDY . . . . . .  L NEAR  0D9D  CODE
XITWRTTY . . . . . .  L NEAR  1406  CODE
XIT_CHKESC . . . . . .  L NEAR  031D  CODE
XIT_PRN_CRLF . . . . . .  L NEAR  1B0C  CODE
XIT_PRTSCR . . . . . .  L NEAR  1ACF  CODE
XIT_TICK_INT . . . . . .  L NEAR  0C6B  CODE
XMIT_@@CON . . . . . .  Number  0069
XMIT_COM1  . . . . . .  Number  0001
XMIT_TIMEOUT . . . . . .  L NEAR  0D48  CODE

YEARLOOP . . . . . .  L NEAR  108F  CODE

ZEROSLOOP  . . . . . .  L NEAR  06EF  CODE

@@CONDSR_RDY . . . . . .  L NEAR  0D21  CODE
@@CONXMIT  . . . . . .  L NEAR  0CFF  CODE
@@SEND . . . . . .  L NEAR  0D29  CODE
@@XMIT_TIME  . . . . . .  L NEAR  0D19  CODE


  4804 Source  Lines
  4938 Total   Lines
   797 Symbols

 14930 Bytes symbol space free

     0 Warning Errors
     0 Severe  Errors
```